Pedro José da Silva Ruivo Antunes
Licenciado

Optimização Estrutural Utilizando Elementos Finitos Híbridos-Trefftz

Dissertação para obtenção do Grau de Mestre
em Engenharia Civil - Perfil Estruturas

Orientador: Prof. Doutor Corneliu Cismasiu, FCT-UNL

Júri:

Presidente: Prof. Doutor Carlos Manuel Chastre Rodrigues
Arguente: Prof. Doutor João Burguete Cardoso
Vogal: Prof. Doutor Corneliu Cismasiu

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro de 2012

Dedicatória

Este trabalho é dedicado ao meu avô, Manuel Antunes.

Agradecimentos

A realização desta dissertação não teria sido possível sem a ajuda de diversas pessoas a quem desde já agradeço do fundo do meu coração. Queria destacar:

O meu orientador, o Prof. Doutor Corneliu Cismasiu, pela oportunidade que me deu, pela disponibilidade e empenho demonstrado ao longo deste trabalho. Queria agradecer igualmente o facto de ter disponibilizado um programa, desenvolvido por si, essencial para a realização deste trabalho.

Aos meus pais, pela educação, carinho e apoio incondicional que me deram ao longo destes anos.
À minha irmã, Maria Inês Antunes, pela sua ternura, paciência e capacidade de me ter conseguido aturar ao longo destes anos.
Às minhas avós, Irene Margarida e Fernanda, pela atenção e carinho.

Aos meus colegas de curso João Grilo (o grande Rafael), Telma Brás, Miguel Ganhão, Pedro Ribeiro, Hélder Parreira, Pedro Rebelo, Linete Afonso e Ana de Sousa pela amizade e apoio nestes anos.
Aos meus amigos, especialmente ao Pedro Gomes, à Joana Fortuna, ao Rui Pereira, à Susana Sá, à Marta Barreto, ao Almiro Gaspar Marques, ao Carolino Ribeirinha, à Brízida e ao Lopes por me terem acompanhado durante estes anos.

Resumo

O progressivo avanço tecnológico tem permitido o estudo aprofundado de temas ligados à engenharia. Se até há poucos anos a resolução de grandes problemas levava muito tempo e requeria um esforço elevado, o aparecimento dos computadores permitiu a sua resolução de uma forma mais rápida e com menor esforço. Actualmente são raros os ramos de Engenharia que não utilizam os computadores como ferramenta de cálculo.

Hoje em dia existem muitos softwares disponíveis para a análise estrutural, que permitem aos seus utilizadores um variado leque de opções, entre as quais: a optimização de estruturas.

Com a optimização estrutural pretende-se tirar o melhor partido possível das estruturas. Para tal é definido um objectivo, um conjunto de exigências que as estruturas terão de respeitar e um conjunto de variáveis que podem sofrer alteração de valor ao longo da optimização.

Apesar do bom desempenho dos elementos finitos híbridos-Trefftz, esta formulação não é muito utilizada. Verifica-se que a maioria dos programas de cálculo de análise estrutural continuam a usar a formulação convencional dos elementos finitos para a resolução de problemas de engenharia.

Tendo presente a observação exposta no parágrafo anterior, este trabalho procura desenvolver um programa que proceda à optimização de forma de estruturas, utilizando elementos finitos híbridos-Trefftz. No final são resolvidos alguns problemas teste e comparados os resultados com os obtidos por outros autores.

Palavras chave:

Optimização estrutural, Optimização de forma, Elementos finitos híbridos-Trefftz, NLPQL

Abstract

The progressive technological breakthrough allows the human being the increasingly in-depth study of issues related to engineering. If, until recently, solving large problems required dispensing methods associated to huge computational effort, the trivialization of performant computers allows nowadays to solve then problems quickly and with less user effort. Today, rare are the engineering branches that do not use computers in their activity.

This technological development allowed the Structural Engineering to predict the behavior of the structures even before their realization. Many software is currently available dedicated to structural analysis, allowing the users to analyse a wide variety of options. The ability to optimize structures is one of them.

Structural optimization means to try to make the best use of the structures. To do this, one have to set an objective, a set of requirements that structures must respect and a set of variables that can change in value over optimization.

Although the good performance of hybrid-Trefftz finite element formulation it, is not widely used and falls against the conventional finite element method that continue to be more used in problem solving engineering software.

Regarding this last point, this work seeks to develop a shape optimization program using the hybrid-Trefftz finite element formulation, thus taking advantage of the potential displayed by this type of elements. Benchmark tests are used to compare the results.

Keywords:

Structural optimization, Shape optimization, Hybrid-Trefftz finite elements method, NLPQL

Índice de Matérias

Índice de Figuras

Índice de Tabelas

Lista de abreviaturas, siglas e símbolos

Abreviaturas

Div	Número de divisões
HTD	Hybrid-Trefftz Displacement method
KKT	Condições Karush-Kuhn-Tucke
Max	Máximo
Min	Mínimo
NLPQL	Non-Linear Programming by Quadratic Lagrangian

Símbolos

Letras maiúsculas latinas

A	Área
$\mathbf{B}$	Matriz Hessiana associada à função de Lagrange
$\mathbf{D}$	matriz operadora diferencial
E	Módulo de elasticidade
I	Momento de inércia
K	Matriz de rigidez
L	Comprimento
$\mathcal{L}$	Função de Lagrange
N	Graus de liberdade
$\mathbf{T}$	Matriz de aproximação das Tracções
$\mathbf{U}_1$	Matriz de aproximação dos deslocamentos - Deformações
$\mathbf{U}_2$	Matriz de aproximação dos deslocamentos - Movimento de corpo rígido

V Volume

V^e Domínio de um corpo

$X(i)$ Variável de projecto i

Letras minúsculas latinas

b Vector das forças de massa

d Vector dos multiplicadores de Lagrange

d_t Grau do polinómio usado na aproximação das tensões

d_u Grau do polinómio usado na aproximação das tensões

f Função objectivo do problema de optimização

g Função restrição do problema de optimização

k Matriz de elasticidade

t Função de aproximação das tracções

u Função de aproximação dos deslocamentos

u_p Solução particular da função de aproximação dos deslocamentos

v Vector resultante da solução do problema de programação quadrática

x Vector das variáveis de projecto

y Vector das variáveis de estado

Letras maiúsculas gregas

Γ^e Fronteira de um corpo

Γ^e_u Fronteira cinemática

Γ^e_σ Fronteira estática

Letras minúsculas gregas

α Indeterminação estática

α_k Comprimento do passo na iteração k

β Indeterminação cinemática

δ Deslocamento

γ Distorção entre os eixos x e y

ε Vector das extensões

ε_r Erro relativo

ϕ Função de mérito

ρ Peso volúmico

σ Vector das tensões

σ^{VM} tensão de Von Mises

σ_{xx} tensão normal segundo a direcção x

σ_{yy} tensão normal segundo a direcção y

τ_{xy} tensão de corte

υ Coeficiente de Poisson

ψ Função de penalização

Capítulo 1

Introdução

1.1 Enquadramento geral

Durante muito tempo os projectos de engenharia basearam-se em cálculos rudimentares ou em conhecimento empírico. Até meados do século XX a realização de um projecto requeria da parte do projectista uma grande experiência. Na altura, pelas mais diversas razões, as decisões eram tomadas com base na intuição ou nas experiências vividas. A falta de recursos e de investimento em equipamento, o desconhecimento de técnicas mais sofisticadas ou o facto de não se dar relevância às tecnologia disponíveis poderão ser as causas mais evidentes.

O desenvolvimento progressivo da área da informática permitiu a aplicação de métodos numéricos. Estes métodos têm vindo a revelar-se como uma mais valia no estudo de problemas relacionados com matemática, engenharia ou física [19]. Técnicas como o método das diferenças finitas ou o métodos dos elementos finitos são agora usadas exaustivamente nos projectos de engenharia. A utilização de computadores na análise estrutural permite, assim, analisar e prever o comportamento das estruturas mesmo antes da sua execução.

O recurso aos computadores foi ganhando força face ao que anteriormente era usado. Este facto permitiu aquilo que intuitivamente o ser humano busca, o óptimo. Nos dias de hoje já não é só relevante projectar uma estrutura que desempenhe a sua função, mas sim, procurar projectar uma estrutura o mais optimizada possível.

Actualmente, existem no mercado vários programas de cálculo disponíveis para a análise estrutural. Na sua maioria, estes programas utilizam na sua análise elementos finitos convencionais. No entanto, alguns estudos indicam que a formulação híbrida-Trefftz apresenta um bom desempenho na resolução deste tipo de problemas. Exemplos destes estudos, em território nacional, são os trabalhos desenvolvidos pelo grupo de investigação de análise estrutural da Universidade Técnica de Lisboa [3].

A crescente comercialização de programas de cálculo de análise estrutural

proporcionou às empresas a procura de mais-valias integrando, assim, mais funcionalidades nos seus produtos. Um deles foi a implementação de optimizações de estruturas. Para tal, o utilizador apenas tem de fornecer os parâmetros referentes à estrutura, qual o objectivo que pretende na sua optimização, as variáveis que quer ter em conta e as restrições que limitam a mesma. Depois, através de processos matemáticos, é obtida a estrutura optimizada.

Existem alguns trabalhos feitos por J.A.Teixeira de Freitas, I.Cismasiu e C.Cismasiu [15, 16] onde se procede à optimização estrutural com elementos finitos híbridos-Trefftz. Porém, a formulação dos elementos finitos convencional continua a ser a mais utilizada nos programas deste tipo.

Dada esta conjuntura, neste trabalho propõe-se desenvolver um programa de cálculo que visa a optimização de forma de estruturas recorrendo a elementos finitos híbridos-Trefftz.

1.2 Objectivos

A presente dissertação tem como objectivo a execução de um programa que possibilita a optimização de forma de estruturas, recorrendo a elementos finitos híbridos-Trefftz. Conjugando um algoritmo de optimização (NLPQL) e um programa de cálculo que utiliza elementos finitos híbridos-Trefftz, pretende-se, desenvolver uma ferramenta que permita resolver problemas relacionados com optimização estrutural.

1.3 Organização da dissertação

A dissertação está organizada em cinco capítulos. Para além deste capítulo introdutório é apresentado no que se segue uma breve descrição dos assuntos abordados nos restantes capítulos:

- **Capítulo 2.** Neste capítulo resume-se a principal informação existente sobre a optimização estrutural e os diferentes tipos existentes. De seguida é apresentado o algoritmo *Non-Linear Programming by Quadratic Lagrangian* (NLPQL) utilizado neste trabalho, sendo o seu desempenho validado através da resolução de dois problemas de teste.
- **Capítulo 3.** No terceiro capítulo é abordado o método dos elementos finitos híbridos-Trefftz. São apresentadas as equações fundamentais deste método assim como o sistema governativo que o rege. Também neste capítulo é explicado o funcionamento do programa utilizado na análise estrutural através do HTD. Por último, apresentam-se alguns exemplos numéricos onde se verifica o seu desempenho.
- **Capítulo 4.** Neste capítulo descreve-se a criação do programa de optimização, o raciocínio utilizado e a sua execução. Por fim são resolvidos

alguns problemas de optimização estrutural recorrendo ao programa desenvolvido. Os resultados são comparados com os obtidos por outros autores e através do software ANSYS.

- **Capítulo 5.** No último capítulo são referidas as conclusões principais a retirar deste trabalho e são apresentados possíveis desenvolvimentos futuros.

Capítulo 2

Optimização Estrutural

Desde sempre a procura pelo óptimo fez parte do quotidiano do Homem, quer seja por razões pessoais ou profissionais. Também nas empresas esta necessidade de alcançar o melhor com os recursos disponíveis é cada vez mais presente. Com o objectivo fulcral de obter maiores lucros, as empresas procuram cada vez mais a prestação de melhores serviços com vista à obtenção de melhores resultados.

Na optimização estrutural, a obtenção do óptimo passa, por exemplo, pela minimização do peso de uma estrutura sem que seja comprometido o seu bom desempenho. Efectivamente, grande parte dos problemas de optimização realizados em estruturas centram-se em minimizar o seu peso ou o seu volume, restringindo valores de tensões ou deformação [26]. A primeira publicação referente à optimização estrutural foi apresentada por Maxwell e remonta a 1890. Utilizando estruturas treliçadas, Maxwell conseguiu provar que é possível obter uma estrutura igualmente funcional mas utilizando menos material. Mais tarde, em 1904, o trabalho de Maxwell foi retomado por Michell [13].

Ao longo dos anos, foram realizadas várias investigações dentro deste âmbito. Surgiram novas técnicas de optimização que juntamente com o desenvolvimento tecnológico têm sido cada vez mais utilizadas em problemas de optimização estrutural. Actualmente existem disponíveis vários programas de cálculo que efectuam optimização de estruturas [18].

2.1 Implementação matemática de um problema de optimização

A um problema de optimização estrutural está sempre associada uma função objectivo (função objectivo) e as respectivas variáveis (variáveis de projecto e de estado) [17, 9]:

- Função objectivo (f): É a função usada para a classificação do projecto. Está associada a um objectivo que poderá passar, por exemplo, por minimizar o peso, maximizar a rigidez de uma estrutura ou ainda minimizar os custos.

- Variáveis de projecto (**x**): Pode representar, por exemplo, a geometria de uma peça ou as características de materiais. São parâmetros que descrevem o projecto e que podem ser alterados durante a optimização.

- Variáveis de estado (**y**): Estas variáveis correspondem à resposta da estrutura. Tornam-se importantes pois podem assegurar a sua fiabilidade no processo de optimização. Às variáveis de estado estão associados valores de mínimo ou máximo para, por exemplo, deformações, tensões ou deslocamentos.

Geralmente, um problema de optimização toma a seguinte forma [9]:

$$(\mathbb{SO}) \begin{cases} \text{Mínimo/Máximo f(}\mathbf{x}\text{) em relação a } \mathbf{x} \\ \text{sujeito a} \begin{cases} \text{restrições de projecto em } \mathbf{x} \\ \text{restrições de comportamento em } \mathbf{y} \\ \text{restrições de equilíbrio} \end{cases} \end{cases}$$

As restrições de projecto estão associadas às variáveis de projecto **x** e as restrições de comportamento associadas às variáveis de comportamento **y**. Estas restrições normalmente são do tipo $g(\mathbf{y}) \leq 0$, onde g representa uma restrição, por exemplo, referente a um deslocamento numa determinada direcção. Estes dois tipos de restrições podem ser facilmente combinados. Por fim, e para os casos de problemas discretos, são ainda implementadas restrições de equilíbrio. Um exemplo deste tipo de restrições poderá ser:

$$\mathbf{F} = \mathbf{Ku} \tag{2.1}$$

Onde **F** representa o vector das forças, **K** a matriz de rigidez e **u** o vector dos deslocamentos.

2.2 Tipos de problemas de optimização estrutural

Os problemas de optimização estrutural podem ser divididos em três grupos consoante a sua tipologia, nomeadamente [9, 5]:

- Optimização dimensional: Este tipo de optimização possibilita determinar as dimensões ideais para a secção transversal dos elementos estruturais, mantendo a topologia e a forma constante. Um exemplo de optimização dimensional está apresentado na Figura 2.1. Com a optimização da estrutura, consegue-se reduzir as dimensões das secções transversais dos elementos estruturais cumprindo as seguintes condições:

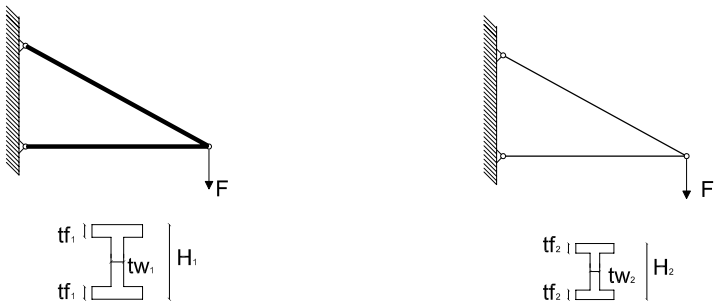
$$H_1 \geq H_2$$

$$tw_1 \geq tw_2$$

$$tf_1 \geq tf_2$$

em que H corresponde à altura da secção, tw à espessura da alma e tf à espessura do banzo.

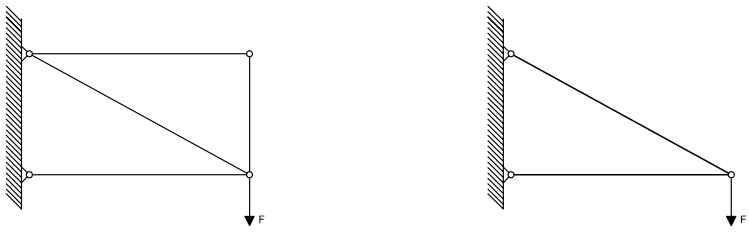
Poderá acontecer que após o processo de optimização todos os elementos estruturais tenham as mesmas dimensões, mas tal facto não é certo, uma vez que as dimensões dos elementos podem variar entre si ou até pode acontecer que nenhuma dessas medidas tenha o mesmo valor.



(a) estrutura inicial (b) estrutura optimizada

Figura 2.1: Problema de optimização dimensional

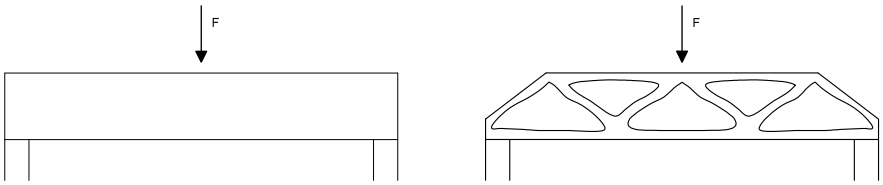
- Optimização topológica: Este tipo de optimização estrutural é o mais genérico. Para casos discretos o objectivo da optimização é reduzir as áreas transversais de elementos pertencentes a estruturas treliçadas, permitindo que o seu valor possa tomar valores nulos, sendo esses elementos estruturais removidos da estrutura. Um exemplo de uma estrutura sujeita a este tipo de optimização é apresentado na Figura 2.2. Em casos de elementos contínuos a optimização, poderá passar por determinar a melhor distribuição do material, podendo introduzir vazios no domínio do elemento. Na Figura 2.3 é apresentado um exemplo deste tipo de optimização que, em alguns casos, passa por uma redução bastante significativa de material.



(a) estrutura inicial (b) estrutura optimizada

Figura 2.2: Problema de optimização topológica de casos discretos (adaptado de [9]).

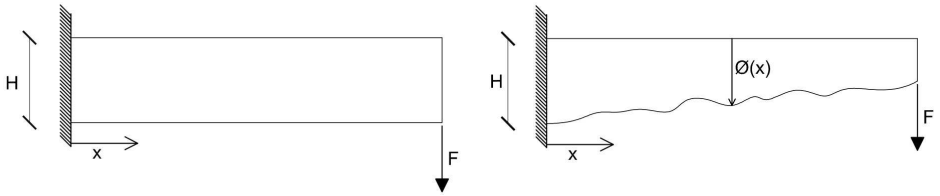
- Optimização de forma: Designada como uma subclasse da optimização topológica, este tipo de optimização mantém apenas a topologia fazendo



(a) estrutura inicial (b) estrutura optimizada

Figura 2.3: Problema de optimização topológica de casos contínuos (adaptado de [9]).

variar a forma. Neste tipo de optimização os contornos externos da estrutura são constituídos por curvas. Pretende-se assim, por exemplo, e conforme a Figura 2.4, uma redução da área da consola através de uma função $\phi(x)$, que define o contorno externo da mesma. Neste tipo de optimização, as variáveis de projecto são os parâmetros referentes às curvas utilizadas na fronteira da consola.



(a) estrutura inicial (b) estrutura optimizada

Figura 2.4: Problema de optimização de forma (adaptado de [9]).

Dos diversos tipos de optimização estrutural descritos, a presente dissertação aprofunda o conhecimento relativo à optimização de forma.

2.3 Métodos numéricos usados em problemas de optimização

Como referido, um problema de optimização é composto por um conjunto de equações. Habitualmente as equações em questão não são triviais e requerem um esforço elevado por parte de quem as resolve. Para facilitar a resolução de problemas de optimização têm vindo a ser utilizados alguns métodos numéricos. Estes, com o auxílio computacional, têm vindo a ser bastantes utilizados no ramo da engenharia [19, 4]. A implementação computacional de métodos numéricos está divida em cinco fases [19]:

1. Escolha do método numérico;
2. Desenvolvimento do algoritmo;

3. Criação do esquema representativo (conhecido por *flow chart*);
4. Programação do algoritmo;
5. Execução do programa para a resolução do problema.

O início do processo passa por escolher um método que possibilite a resolução do problema em causa. Numa segunda fase, elabora-se um conjunto bem definido de instruções sequenciais necessárias para resolução do método, ao qual se dá o nome de algoritmo. Trata-se de uma fase importante pois apenas as instruções definidas no algoritmo serão implementadas computacionalmente. O algoritmo possuirá a informação de quando a máquina deverá começar ou terminar, quais as operações que deverá executar e a respectiva sequência. Na terceira fase é realizada um esquema representativo do algoritmo por forma a facilitar a compreensão do mesmo. Nestes tipos de esquemas, *flow charts*, é utilizada uma simbologia que representa as diversas operações definidas no algoritmo. Neste trabalho irão ser utilizados os símbolos estandardizados que podem ser consultados em [8].

Na quarta fase passa-se à programação do algoritmo definido para a resolução do método utilizando uma das linguagens de programação existentes. Por fim, e de forma a verificar se todo o processo está a funcionar correctamente, são realizados testes.

Actualmente, existem alguns algoritmos para problemas de optimização, com os quais se podem solucionar este tipo de situações. De seguida são apresentados alguns deles:

- Algoritmos Simplex - É uma das ferramentas utilizadas na optimização linear. Caracteriza por resolver problemas em que tanto as restrições como a função objectivo são compostas exclusivamente por funções lineares. Os principais desenvolvimentos teóricos da optimização linear foram devidos a Kantorovich (1939). A formulação do algoritmo simplex é fruto do trabalho de B. Dantzig (1939 a 1951) [12].
- Algoritmos de Ponto Interior - Baseado no método do ponto interior, este algoritmo possibilita resolver problemas compostos por funções não lineares. Foi inicialmente alvo de estudo por volta da década de sessenta, mas o seu uso caiu em desuso face ao aparecimento de novas técnicas mais eficazes como, por exemplo, o método de programação quadrática [14].
- Algoritmos Genéticos - Baseados na teoria da evolução desenvolvida por Darwin (1859) estes algoritmos são utilizados na resolução de problemas de optimização. Inicialmente estes algoritmos foram utilizados na área da genética. Ao longo dos tempos têm vindo a ser aplicado a outras áreas como por exemplo biologia, ciências socais ou, até mesmo, em engenharia estrutural, como se pode ver na investigação de Cardoso [7].
- Algoritmo NLPQL - Recorrendo ao método de programação quadrática sequencial, este algoritmo possibilita resolver problemas de optimização

com funções não lineares. Desenvolvido por K. Schittowski (1981) este algoritmo tem sido usado em bastantes áreas científicas e tem vindo a fazer parte de alguns softwares do ramo da engenharia, como por exemplo: ANSYS (para optimização estrutural), STRUTEL (para análises de confiança) ou ainda TEMPO (para o controlo de centrais eléctricas) [23, 22].

Dos algoritmos apresentados anteriormente, neste trabalho utiliza-se o NLPQL por apresentar resultados bastante satisfatórios conforme se poderá verificar de seguida.

2.4 Algoritmo NLPQL

Desenvolvido por K. Schittowski em 1981, o *Non-Linear Programming by Quadratic Lagrangian* (NLPQL) é um algoritmo que possibilita resolver problemas de optimização recorrendo ao método de programação quadrática sequencial. Este método permite a resolução de problemas de optimização para funções não lineares quer sejam restrições ou a função objectivo. O NLPQL foi testado em cerca de 700 problemas teste que ajudaram a concluir que existiam algumas limitações [23]:

- Todas as funções presentes no problema (sejam restrições ou função objectivo) têm de ser diferenciáveis nos intervalos a que elas estão associadas;

- O problema não pode ser muito extenso. Os testes realizados indicam que o NLPQL funciona correctamente para problemas com menos de 100 variáveis. Ao longo deste trabalho, não será ultrapassado esse valor, de maneira que não existirão este tipo de problemas.

O método de programação quadrática sequencial foi desenvolvido principalmente por S. P. Han e M. J. D. Powell e baseado no trabalho de R. B. Wilson [23]. Este método consiste em transformar o problema inicial num subproblema de programação quadrática e resolvê-lo [22].

Tomando em conta o problema genérico de optimização [23]:

$$\begin{aligned} & Min \;\; f(\mathbf{x}) \\ x \in \Re^n : \;\; & g_j(\mathbf{x}) = 0 \quad j = 1, ..., m_e \\ & g_j(\mathbf{x}) \geq 0 \quad j = m_e, ..., m \\ & \mathbf{x}_q \leq \mathbf{x} \leq \mathbf{x}_u \end{aligned} \tag{2.2}$$

Consideremos $\mathbf{x}_k$ como o vector a que estão associados os valores das variáveis na iteração k, $\mathbf{v}_k$ o vector correspondente à aproximação dos multiplicadores de Lagrange óptimos da iteração k e $\mathbf{B}_k$ a aproximação à matriz Hessiana na iteração

k associada à função de Lagrange:

$$\mathcal{L}(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) - \sum_{j=1}^{m'} \mathbf{u}_j g_j(\mathbf{x}) \tag{2.3}$$

onde $\mathbf{x} \in \Re^n$, $\mathbf{u} \in \Re^{m'}$ e $m' = m + 2n$.

Atendendo que se pode re-escrever as restrições de fronteira ($\mathbf{x}_q \leq \mathbf{x} \leq \mathbf{x}_u$) em duas novas restrições para cada uma das variáveis:

$$\begin{aligned} g_j(\mathbf{x}) &= x^{j-m} - x_q^{(j-m)} \quad j = m+1, ..., m+n \\ g_j(\mathbf{x}) &= x_u^{j-m-n} - x^{(j-m)} \quad j = m+n+1, ..., m+m' \end{aligned} \tag{2.4}$$

Linearizando as restrições não lineares de (2.2) e minimizando a equação de Lagrange (2.3) obtém-se o subproblema de programação quadrática [23]:

$$\begin{aligned} & Min \ \frac{1}{2}\mathbf{d}^T \mathbf{B}_k \mathbf{d} + \nabla f(\mathbf{x}_k)^T \mathbf{d} \\ \mathbf{d} \in \Re^n : \quad & \nabla g_j(\mathbf{x}_k)^T \mathbf{d} + g_j(\mathbf{x}) = 0 \quad j = 1, ..., m_e \\ & \nabla g_j(\mathbf{x}_k)^T \mathbf{d} + g_j(\mathbf{x}) \geq 0 \quad j = m_e + 1, ..., m \\ & \mathbf{x}_q - \mathbf{x}_k \leq \mathbf{d} \leq \mathbf{x}_u - \mathbf{x}_k \end{aligned} \tag{2.5}$$

Do subproblema (2.5) resulta o vector $\mathbf{d}_k$ que juntamente com o vector $\mathbf{u}_k$, que corresponde ao vector dos multiplicadores de Lagrange do subproblema de programação quadrática, possibilitam determinar os novos valores do vector **x** e **v** a considerar na iteração seguinte:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{d}_k \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \alpha_k (\mathbf{u}_k - \mathbf{v}_k) \end{aligned} \tag{2.6}$$

O parâmetro α_k apresentado em (2.6) corresponde ao comprimento do passo, com o qual se pretende reduzir a função de mérito:

$$\phi_k(\alpha) = \psi r_k \left[\begin{pmatrix} \mathbf{x}_k \\ \mathbf{v}_k \end{pmatrix} + \alpha \begin{pmatrix} \mathbf{d}_k \\ \mathbf{u}_k - \mathbf{v}_k \end{pmatrix} \right] \tag{2.7}$$

À expressão (2.7) está associado uma função de penalização ψr_k que controla o grau de penalização da função objectivo ou da função de Lagrange quando deixam a região admissível. Uma possível escolha para ψr_k pode passar pela função exacta [23]:

$$\psi r_k(x, v) = f(\mathbf{x}) + \sum_{j=1}^{m_e} r_j \left| g_j(\mathbf{x}) \right| + \sum_{j=m_e+1}^{m'} r_j \left| min(0, g_j(\mathbf{x})) \right| \tag{2.8}$$

ou a função de Lagrange aumentada [23]:

$$\psi r_k(x,v) = f(\mathbf{x}) - \sum_{j=1}^{m_e}\left(\mathbf{v}_j g_j(\mathbf{x}) - \frac{1}{2} r_j g_j(\mathbf{x})^2\right) \tag{2.9}$$

$$- \sum_{j=m_e+1}^{m'} \begin{cases} \mathbf{v}_k g_j(\mathbf{x}) - \frac{1}{2} r_j g_j(\mathbf{x})^2; & se \quad g_j(\mathbf{x}) \leq \frac{\mathbf{v}_j^2}{r_j} \\ \frac{1}{2}\frac{\mathbf{v}_j^2}{r_j}; & se \quad g_j(\mathbf{x}) > \frac{\mathbf{v}_j^2}{r_j} \end{cases}$$

Nas expressões anteriores r_j corresponde a um parâmetro de penalização adequado para garantir o decréscimo da função de mérito. Contudo, nem sempre é possível implementar o subproblema de programação quadrática. É possível que a região admissível de (2.5) esteja vazia, embora o problema original (2.2) seja resolúvel. Para evitar problemas é introduzido uma nova variável, δ [22]:

$$\begin{aligned} & Min \ \frac{1}{2}\mathbf{d}^T \mathbf{B}_k \mathbf{d} + \nabla f(\mathbf{x}_k)^T \mathbf{d} + \frac{1}{2}\rho_k \delta^2 \\ \mathbf{d} \in \Re^n, \delta \in \Re : \quad & \nabla g_j(\mathbf{x}_k)^T \mathbf{d} + (1-\delta) g_j(\mathbf{x}) \left\{ \begin{array}{c} = \\ \geq \end{array} \right\} 0 \quad j \in J_k \\ & \nabla g_j(\mathbf{x}_k)^T \mathbf{d} + g_j(\mathbf{x}) \geq 0 \quad j \in K_k \\ & \mathbf{x}_q - \mathbf{x}_k \leq \mathbf{d} \leq \mathbf{x}_u - \mathbf{x}_k \\ & 0 \leq \delta \leq 1 \end{aligned} \tag{2.10}$$

onde

$$J_k = \{1, ..., m_e\} \cup \left\{ j : m_e < j \leq m, g_j(\mathbf{x}_k) \leq \epsilon \ \ ou \ \ v_j^k > 0 \right\}$$

$$K_k = \{1, ..., m\} / J_k$$

O termo ρ_k corresponde a um parâmetro penalizador adicional com o qual se pretende reduzir a influencia da variável δ na solução do subproblema (2.10). Este processo finaliza quando as condições de Karush Kuhn-Tucker (KKT) são verificadas. Estas condições podem ser consultadas no **apêndice A**.

O algoritmo utilizado no NLPQL traduz o processo atrás mencionado. Para tal, o NLPQL utiliza um conjunto de subrotinas que processam o cálculo da solução óptima do problema. De seguida são apresentadas as subrotinas e a forma como se encontram articuladas entre si. Recorreu-se ao fluxograma apresentado na Figura 2.5.

O programa de cálculo começa por efectuar a leitura de informação relativa ao número de variáveis e de restrições associadas ao problema. É também neste ficheiro que se encontram registados os valores arbitrados para as variáveis do problema. Depois de definir estes parâmetros é executada a subrotina NLPQL.

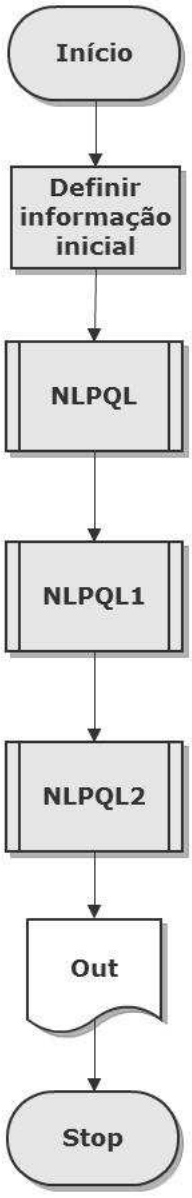


Figura 2.5: Fluxograma representativo da organização do programa NLPQL.

A subrotina NLPQL tem como principal objectivo transformar o problema da sua forma inicial (2.2) no subproblema de programação quadrática (2.5), evitando assim adicionar a variável δ ao subproblema. Caso não seja possível, a subrotina insere a variável δ no subproblema conforme apresentado em (2.10) [23].

De seguida, e através da subrotina NLPQL1, é possível obter as estimativas iniciais para os multiplicadores e para a matriz Hessiana de Lagrange. Trata-se de um processo que não é controlado pelo utilizador. Porém, caso o deseje o utilizador poderá fornecer estes parâmetros. Para tal é igualmente nesta subrotina que o poderá fazer [23]. Neste trabalho o cálculo dos multiplicadores de Lagrange e da matriz Hessiana foi feita sem influência do utilizador.

A subrotina NLPQL2 é responsável por grande parte da resolução do problema sequencial de programação quadrática. Na Figura 2.6 é apresentado o fluxograma desta subrotina.

Esta subrotina começa por calcular os valores da função objectivo, das restrições e das derivadas parciais de todas as funções associadas ao problema. Para tal procede à leitura das subrotinas NLFUNC e NLGRAD. Consideremos o problema de optimização:

$$\begin{aligned} Min \;\; & f(x_1, x_2) = x_1 + x_2 \\ & g_j(x_1, x_2) = x_1^2 - x_2^2 \geq 0 \\ & 0 \leq x_1 \leq 100 \\ & 0 \leq x_2 \leq 500 \end{aligned} \tag{2.11}$$

Pretende-se com a subrotina NLFUNC que a cada iteração, sejam obtidos os valores correspondentes às funções do problema. De acordo com o exemplo anterior, se considerarmos que as variáveis à entrada para a subrotina tomam o valor de 1 para a x_1 e 3 para x_2 no final da leitura, a subrotina apenas remeterá o valor da função objectivo, que neste caso é $f(x_1, x_2) = 4$ e o valor da restrição, $g(x_1, x_2) = -8$. Esta subrotina é desenvolvida pelo utilizador e nela apenas deverá conter as funções integrantes do problema.

O mesmo procedimento é aplicado à subrotina NLGRAD, onde o utilizador fornece as derivadas parciais das funções associadas ao problema. Assim, e mais uma vez, utilizando como exemplo o problema (2.11) tem-se:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 1 \qquad \frac{\partial f(x_1, x_2)}{\partial x_2} = 1$$

$$\frac{\partial g(x_1, x_2)}{\partial x_1} = 2x_1 \qquad \frac{\partial g(x_1, x_2)}{\partial x_2} = -2x_2$$

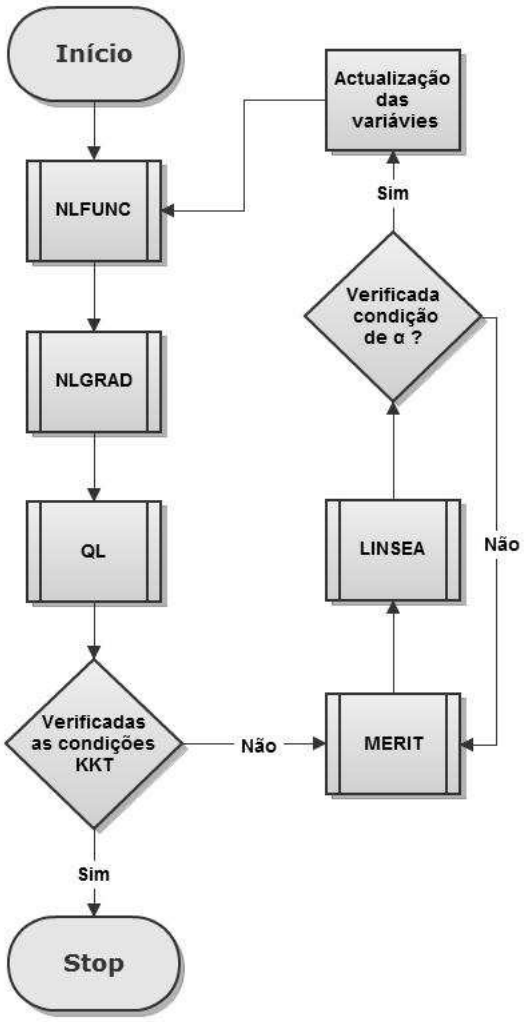


Figura 2.6: Fluxograma representativo da organização da subrotina NLPQL2.

Relembrando que $x_1 = 1$ e $x_2 = 3$, no final da leitura da subrotina apenas os valores das funções atrás mencionadas serão utilizados, sendo:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 1 \qquad \frac{\partial f(x_1, x_2)}{\partial x_2} = 1$$
$$\frac{\partial g(x_1, x_2)}{\partial x_1} = 2 \qquad \frac{\partial g(x_1, x_2)}{\partial x_2} = -6$$

Posteriormente, e com os valores obtidos através do NLFUNC e NLGRAD, é resolvido o subproblema de programação quadrática através da subrotina QL. De notar que a conversão do problema foi realizada anteriormente através da subrotina NLPQL.

Após a resolução do subproblema, no caso de todas as condições KKT estarem verificadas, a subrotina pára e o processo de optimização termina. Caso contrário começam a ser calculados os valores que as variáveis devem tomar na próxima iteração.

Para o cálculo do valor das variáveis é necessário determinar o comprimento do passo α. Conforme referido em [22], minimizar a função de mérito (2.7) de forma a obter o valor de α não é possível. Assim, o programa NLPQL estabelece que o parâmetro α terá de verificar a condição:

$$\phi_r(\alpha_i) < \phi_r(0) + \mu\alpha_i\phi_r^{'}(0) \tag{2.12}$$

Onde $0 < \beta < 1$ e $0 < \mu < 0.5$.

A partir da subrotina MERIT, começa-se por calcular os valores para a função de mérito ϕ_r. Na primeira iteração α toma valor unitário. Esta subrotina está programada para resolver a função de Lagrange aumentada (2.9), mas pode ser utilizada outra, como por exemplo a função exacta (2.8). Para tal o utilizador terá de a fornecer [23]. Neste trabalho utilizou-se a função pré-definida.

Posteriormente, a verificação da condição (2.12) é feita através da subrotina LINSEA. No caso de a condição não estiver verificada o valor de α toma o valor expresso em (2.13) e volta a cálcular o valor de $\phi_r(\alpha_i)$ através da subrotina MERIT.

$$\alpha_{i+1} = Max\left[\beta\alpha_i, \left(0.5\alpha_i^2\phi_r^{'}(0)\right) / \left(\alpha_i\phi_r^{'}(0) - \phi_r(\alpha_i) + \phi_r(0)\right)\right] \tag{2.13}$$

Este processo finaliza quando se obtem um valor de α que verifique a condição (2.12).

Por fim, são actualizadas as variáveis do problemas e os multiplicadores de

Lagrange como apresentado em (2.6) e retoma-se o cálculo do subproblema até serem verificadas as condições KKT. Após todo o processo, os resultados obtidos podem ser consultados no ficheiro Out.

Importa referir que o utilizador para executar com sucesso o programa, ao qual está associado o algoritmo NLPQL, tem de desenvolver duas subrotinas (a NLFUNC e a NLGRAD) e o Main Program. No Main Program o utilizador tem de fornecer algumas informações relativas ao problema e no final "chamar"a subrotina NLPQL, a partir da qual começa o processo de cálculo descrito anteriormente.

De seguida serão resolvidos dois problemas de optimização estrutural de forma a analisar o desempenho do NLPQL.

2.4.1 Problema de minimização do peso de uma consola

O objectivo deste problema é minimizar o peso de uma consola, sem que o deslocamento na sua extremidade ultrapasse o valor estabelecido δ_0. A resolução deste problema é apresentado em [9]. A viga é constituída por N segmentos de igual comprimento L. O primeiro elemento da consola encontra-se na extremidade livre e o último (elemento N) na extremidade encastrada. A espessura t da secção mantêm-se constante para todos os elementos ao contrário das dimensões. Trata-se de uma secção quadrada e vazada que varia a sua dimensão x_i consoante o elemento, conforme apresentado na Figura 2.7.

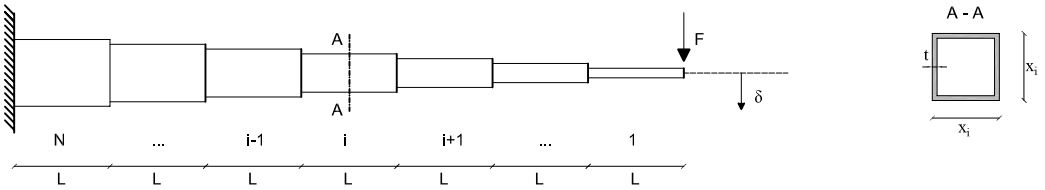


Figura 2.7: Viga em consola composta po N elementos e sujeita a um carregamento vertical F na extremidade (adaptado de [9]).

A definição da função objectivo será uma expressão que traduz o peso da estrutura e que pode ser calculada com a seguinte expressão:

$$f(x_1, ..., x_N) = \sum_{i=1}^{N} (\rho L[x_i^2 - (x_i - 2t)^2] = \rho L \sum_{i=1}^{N} [x_i^2 - (x_i - 2t)^2] \tag{2.14}$$

onde ρ representa a densidade do material e como foi referido anteriormente L , t e x_i é o comprimento do elemento, a espessura e a largura da secção respectivamente. Assumindo que a espessura t é muito inferior à largura x_i de cada secção, a expressão (2.14) pode ser simplificada para:

$$f(x_1, ..., x_N) = 4\rho L t \sum_{i=1}^{N} x_i \tag{2.15}$$

Ao definir $\delta^{(i)}$ o deslocamento na ponta da consola quando apenas o segmento i é elástico e os demais rígidos e, admitindo válido o principio da sobreposição de efeitos, o deslocamento pode ser definido pelo somatório do contributo de cada elemento:

$$\delta = \sum_{i=1}^{N} \delta^{(i)} \tag{2.16}$$

Recorrendo à Figura 2.8 e, admitindo a hipótese dos pequenos deslocamentos, o deslocamento para um dos N elementos pertencentes à consola pode ser expresso por:

$$\delta^{(i)} = \delta_i + (i-1)L\theta_i \tag{2.17}$$

em que δ_i e θ_i são o deslocamento e a rotação na extremidade direita do segmento i.

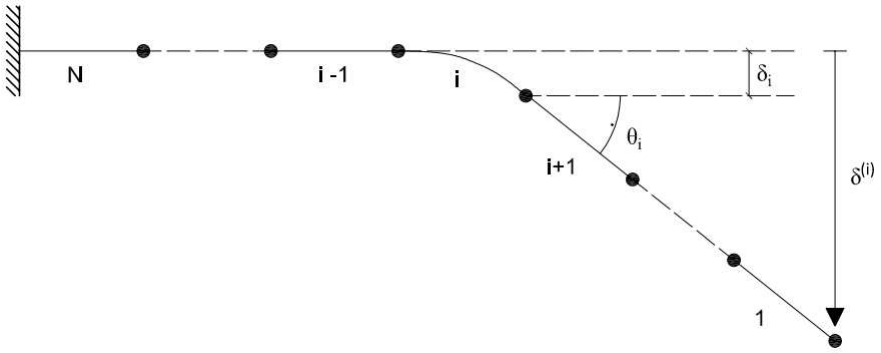


Figura 2.8: Esquema representativo do cálculo do deslocamento $\delta^{(i)}$

O valor do deslocamento δ_i e da rotação θ_i pode ser obtido, por exemplo, através do método dos deslocamentos admitindo uma barra encastrada. Essa barra está sujeita na extremidade livre uma força concentrada F_i e um momento M_i conforme a Figura 2.9. O momento flector e o esforço transverso numa secção i podem ser obtidos por equilíbrio e apresentam os seguintes resultados:

$$M_i = (i-1)FL; \qquad F_i = F \tag{2.18}$$

A equação do métodos dos deslocamentos particulariza-se neste caso como:

$$\begin{bmatrix} \frac{12EI_i}{L^3} & \frac{-6EI_i}{L^2} \\ \frac{-6EI_i}{L^2} & \frac{4EI_i}{L} \end{bmatrix} \begin{bmatrix} \delta_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} F_i \\ M_i \end{bmatrix} \tag{2.19}$$

Substituindo em (2.19) os resultados obtido do equilibrio de esforços em (2.18) obtem-se as expressões para o deslocmaento δ_i e a rotação θ_i:

$$\begin{bmatrix} \frac{12EI_i}{L^3} & \frac{-6EI_i}{L^2} \\ \frac{-6EI_i}{L^2} & \frac{4EI_i}{L} \end{bmatrix} \begin{bmatrix} \delta_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} F \\ (i-1)FL \end{bmatrix} \tag{2.20}$$

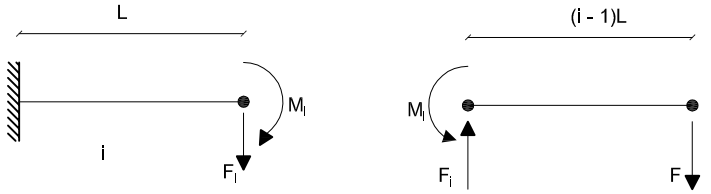


Figura 2.9: Representação de um segmento i da consola, com os respectivos esforços.

$$\begin{cases} \delta_i = \frac{FL^3}{3EI_i} + \frac{(i-1)FL^3}{2EI_i} \\ \theta_i = \frac{FL^2}{2EI_i} + \frac{(i-1)FL^2}{EI_i} \end{cases} \tag{2.21}$$

O deslocamento na extremidade livre da consola resulta da substituição dos resultados (2.21) na expressão (2.17):

$$\delta^{(i)} = \frac{FL^3}{EI}\left(i^2 - i + \frac{1}{3}\right) \tag{2.22}$$

Calculando o momento de inércia:

$$I_i = \frac{x_i^4}{12} - \frac{(x_i - 2t)}{12} = \frac{2tx_i^3}{3} \tag{2.23}$$

Obtem-se a expressão o deslocamento total:

$$\delta = \frac{3FL^3}{2tE}\sum_{i=1}^{N}\left(i^2 - i + \frac{1}{3}\right)\frac{1}{x_i^3} \tag{2.24}$$

A restrição do problema de optimização estabelecido correponde a $\delta \leq \delta_0$, assim a expressão correspodente pode ser escrita como:

$$\sum_{i=1}^{N}\left(i^2 - i + \frac{1}{3}\right)\frac{3}{x_i^3} \leq \frac{2tE}{FL^3}\delta_0 \tag{2.25}$$

Por fim, pode-se definir o problema de optimização para um número genérico N de elementos. Para tal, basta definir a expressão 2.15 como sendo a função objectivo e a expressão 2.25 como uma das restrições. As restantes restrições baseam-se em definir que o comprimento x_i para cada secção seja sempre maior que zero. Assim, e recorrendo à estrutura pré definida anteriormente:

$$(\mathbb{SO}) \begin{cases} Min \;\; f(x_1,...,x_N) = C_1 \sum_{i=1}^{N} x_i \\ \text{Sujeito a} \begin{cases} \sum_{i=1}^{N} \left(i^2 - i + \frac{1}{3}\right) \frac{3}{x_i^3} \leq C_2 \\ x_N > 0 \end{cases} \end{cases} \tag{2.26}$$

onde, $C_1 = 4\rho Lt$ e $C_2 = 2tE\delta_0/(FL^3)$.

Admitindo que a consola apresenta dois elementos (N=2), pode-se obter a solução analítica do problema de optimização. Neste caso o problema define-se por:

$$(\mathbb{SO}) \begin{cases} Min \;\; f(x_1,x_2) = C_1(x_1 + x_2) \\ \text{Sujeito a} \begin{cases} \frac{1}{x_1^3} + \frac{7}{x_2^3} \leq C_2 \\ x_1 > 0 \\ x_2 > 0 \end{cases} \end{cases} \tag{2.27}$$

Mantendo as mesmas definições de C_1 e C_2. Para que seja possível a resolução analítica deste problema terá de se relacionar a função objectivo com as restrições a que está sujeita. Assim, recorrendo à restrição $1/(x_1^3) + 7/(x_2^3) \leq C_2$, pode-se definir x_2, como $x_2 = 7^{\frac{1}{3}} x_1/(x_1^3 C_2 - 1)^{\frac{1}{3}}$ e substituir-se na função objectivo:

$$f(x_1) = C_1 \left(x_1 + \frac{7^{\frac{1}{3}} x_1}{(x_1^3 C_2 - 1)^{\frac{1}{3}}} \right) \tag{2.28}$$

Fazendo a função objectivo apenas em função de um das variáveis, o mínimo da função pode ser calculado derivando a função objectivo e igualando-a a zero. Para se saber se esta variável tem como valor óptimo um valor positivo, respeitando assim a restrição de que todas as variáveis terão valor maior de zero, pode-se calcular a segunda derivada da função objectivo e verificar-se nesse ponto é positiva, o que efectivamente se verifica para este caso. Assim o valor óptimo para x_1 será:

$$\frac{\partial f(x_1)}{\partial x_1} = C_1 - \frac{7^{\frac{1}{3}} C_1}{\left(x_1^3 C_2 - 1\right)^{\frac{4}{3}}} = 0 \;\; \Rightarrow \;\; x_1^* = \left(\frac{7^{\frac{1}{4}} + 1}{C_2} \right)^{\frac{1}{3}} \tag{2.29}$$

Procedendo com o mesmo raciocínio para a determinação do valor óptimo para x_2 resulta que $x_1 = x_2/\left(C_2 x_2^3 - 7\right)^{\frac{1}{3}}$ e, substituindo na função objectivo, derivando-a e determinando o valor óptimo de x_2 obtêm-se:

$$\frac{\partial f(x_2)}{\partial x_2} = C_1 \left[1 - \frac{7}{(C_2 x_2^3 - 7)^{\frac{4}{3}}} \right] \Rightarrow x_2^* = 7^{\frac{1}{4}} \left(\frac{1 + 7^{\frac{1}{4}}}{C_2} \right)^{\frac{1}{3}} \tag{2.30}$$

Resumindo, os valores óptimos para o caso em que a viga consola tenha 2 elementos são os seguintes:

$$\begin{cases} x_1^* = \left(\frac{7^{\frac{1}{4}}+1}{C_2} \right)^{\frac{1}{3}} \\ x_2^* = 7^{\frac{1}{4}} \left(\frac{1+7^{\frac{1}{4}}}{C_2} \right)^{\frac{1}{3}} \\ f(x_1^*, x_2^*) = \frac{(7^{\frac{1}{4}}+1)^{\frac{4}{3}}}{C_2^{\frac{1}{3}}} C_1 \end{cases} \tag{2.31}$$

Recorde-se que $C_1 = 4\rho L t$ e $C_2 = 2tE\delta_0/(FL^3)$.

No decorrer deste trabalho foram ainda calculados os resultados analíticos para o caso de $N = 3, 5, 10$. Estes resultados podem ser vistos no **Apêndice A**.

Para ser possível a comparação entre os resultados obtidos com o algoritmo de optimização e os resultados analíticos, arbitraram-se os seguintes valores para os parâmetros característicos da estrutura:

$$\begin{aligned} \rho = 7850\ kg/m^3 \qquad & L = 2\ m \\ t = 0.05\ m \qquad & E = 200\ Gpa \\ F = 450\ kN \qquad & \delta = 0.02\ m \end{aligned}$$

Possibilitando o calcular as constantes $C1$ e $C2$, que tomam os valores de 3140 kg/m e 111.11 m^{-3}, respectivamente. Os resultados obtidos, para uma divisão da consola em 2 elementos, são apresentados na Tabela 2.1.

Nº Secções	Solução analítica (kg)	NLPQL		MATLAB	
		Nº iterações	Peso (kg)	Nº iterações	Peso (kg)
2	2367.0	14	2366.9	15	2367.0

Tabela 2.1: Optimização do peso da consola - Resultados obtidos.

Como se pode verificar, o algoritmo NLPQL atinge quase por completo o resultado obtido analiticamente provando, assim, que se consegue obter um bom resultado utilizando este algoritmo. Na tabela também se encontram registados os resultados obtidos através de um outro algoritmo disponível no software MATLAB.

Trata-se do algoritmo *Interior-Point Algorithm* e, como foi apresentado em 2.3, é baseado no método do ponto interior. De seguida, procedeu-se à realização de um gráfico onde se pode observar a evolução do erro relativo ao longo das iterações que cada algoritmo necessitou. O erro relativo foi determinado através da seguinte expressão [19]:

$$\varepsilon_r = \frac{|\overline{x} - x|}{|\overline{x}|} \times 100\% \tag{2.32}$$

Onde $\overline{x}$ corresponde ao valor analítico e x ao valor aproximado.

O gráfico representativo da evolução do erro relativo é apresentado na Figura 2.10.

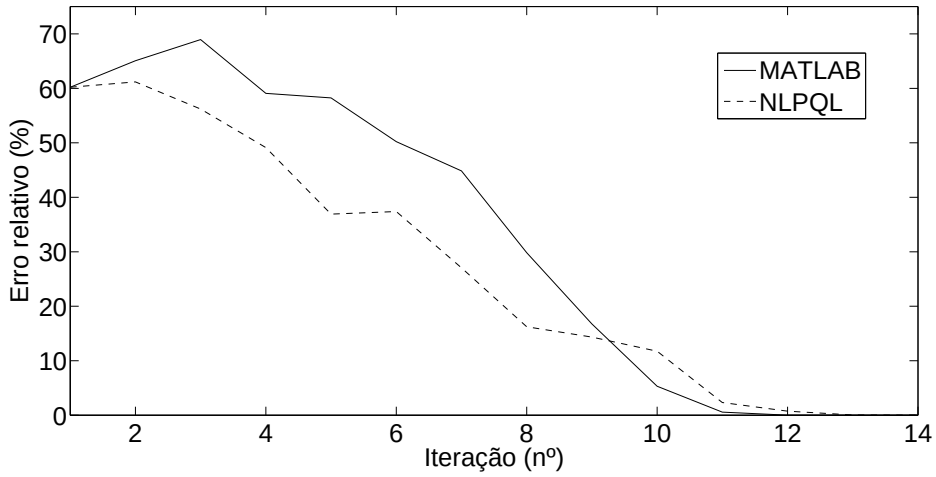


Figura 2.10: Evolução do erro durante o processo de optimização.

Pelo gráfico verifica-se que o algoritmo NLPQL, nas primeiras iterações, converge mais rapidamente, deixando-se depois ultrapassar por volta da nona iteração quando os erros tomam valores aproximados de 15%. Contudo e, tendo em conta que na décima terceira iteração os erros obtidos são praticamente nulos, pode-se concluir que o algoritmo NLPQL obteve um bom desempenho. À medida que o erro vai diminuindo, o algoritmo NLPQL tende a ser ultrapassado pelo algoritmo utilizado no MATLAB. Porém o principal objectivo da realização deste problema fica atingido. O algoritmo NLPQL procede à resolução dos problemas e fá-lo de forma bastante satisfatória. Analisando os casos em que a consola é constituída por mais que dois elementos (apresentados no **apêndice A**) pode-se concluir que, com o aumento de variáveis em jogo na optimização, o algoritmo utilizado pelo software Matlab apresenta mais dificuldade em se aproximar dos resultados analíticos. O mesmo também parece acontecer para o algoritmo NLPQL. Contudo, este aproxima-se mais rapidamente que o outro algoritmo em estudo. Os códigos utilizados nas subrotinas NLFUNC,NLGRAD e no MAIN PROGRAM podem ser

consultados no **apêndice B**.

2.4.2 Problema de maximização da rigidez de uma estrutura treliçada

O problema apresentado encontra-se na bibliografia [9] e centra-se na maximização da rigidez através da minimização do trabalho das forças aplicadas, sem que para tal seja ultrapassado o valor V_0 para o volume da treliça. A estrutura é composta por 3 barras sobre a qual actua uma força P conforme a Figura 2.11. As variáveis presentes no problema são os valores da área de cada uma das barras (A_1, A_2, A_3). Pretende-se, assim, resolver um problema com a seguinte tipologia:

$$(\mathbb{SO}) \begin{cases} Min \;\; \mathbf{F}^T\mathbf{u}(x) \\ \text{Sujeito a} \begin{cases} \mathbf{L}^T x \leq \mathbf{V}_0 \\ x \in \Re^+ \end{cases} \end{cases}$$

Em que **F** representa o vector das forças nodais, **u** os delocamentos nodais, **L** os comprimentos das barras e $\mathbf{V}_0$ ao volume máximo das treliças.

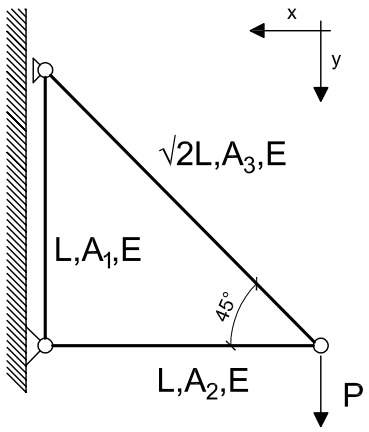


Figura 2.11: Estrutura treliçada composta por 3 barras sujeita á minimização da flexibilidade

Para o cálculo da função objectivo, de forma a determinar os deslocamentos pode-se recorrer os método dos deslocamentos. Neste caso tem-se de primeiramente proceder ao cálculo da matriz de rigidez (**K**) segundo os graus de liberdade indicados na Figura 2.12.

Assim a matriz **K** é:

$$\mathbf{K} = \frac{E}{L} \begin{bmatrix} A_2 + \frac{A_3}{2\sqrt{2}} & -\frac{A_3}{2\sqrt{2}} & \frac{A_3}{2\sqrt{2}} \\ -\frac{A_3}{2\sqrt{2}} & \frac{A_3}{2\sqrt{2}} & -\frac{A_3}{2\sqrt{2}} \\ \frac{A_3}{2\sqrt{2}} & -\frac{A_3}{2\sqrt{2}} & A_1 + \frac{A_3}{2\sqrt{2}} \end{bmatrix} \tag{2.33}$$

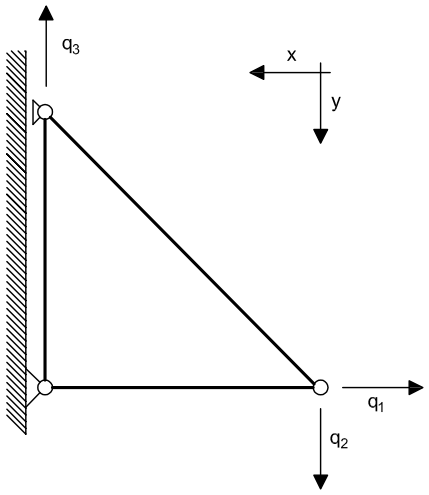


Figura 2.12: Graus de liberdade cinemáticos da estrutura treliçada

Conhecida a matriz de rigidez (**K**) e recorrendo à equação do método dos deslocamentos, determina-se o vector dos deslocamentos nodais (**u**):

$$\begin{bmatrix} 0 \\ P \\ 0 \end{bmatrix} = \begin{bmatrix} A_2 + \frac{A_3}{2\sqrt{2}} & -\frac{A_3}{2\sqrt{2}} & \frac{A_3}{2\sqrt{2}} \\ -\frac{A_3}{2\sqrt{2}} & \frac{A_3}{2\sqrt{2}} & -\frac{A_3}{2\sqrt{2}} \\ \frac{A_3}{2\sqrt{2}} & -\frac{A_3}{2\sqrt{2}} & A_1 + \frac{A_3}{2\sqrt{2}} \end{bmatrix} . \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \Leftrightarrow \tag{2.34}$$

$$\Leftrightarrow \mathbf{u} = \frac{PL}{E} \begin{bmatrix} \frac{1}{A_2} \\ \frac{1}{A_1} + \frac{1}{A_2} + \frac{2\sqrt{2}}{A_3} \\ \frac{1}{A_1} \end{bmatrix} \tag{2.35}$$

Visto que é conhecido o vector dos deslocamentos (**u**) e o vector das forças nodais (**F**) pode-se substituir na função objectivo:

$$f(A_1, A_2, A_3) = \mathbf{F}^T\mathbf{u} = C_1 \left(\frac{1}{A_1} + \frac{1}{A_2} + \frac{2\sqrt{2}}{A_3} \right) \tag{2.36}$$

onde $C_1 = P^2 L/E$.

Por outro lado o problema está sujeito á condição:

$$\mathbf{L}^T A \leq \mathbf{V}_0 \Leftrightarrow L\left(A_1 + A_2 + \sqrt{2}A_3\right) - V_0 \leq 0 \tag{2.37}$$

Pelo que, o problema de optimização fica definido pelo sistema:

$$(\mathbb{SO}) \begin{cases} Min \;\; f(A_1, A_2, A_3) = F^T u = C_1 \left(\frac{1}{A_1} + \frac{1}{A_2} + \frac{2\sqrt{2}}{A_3} \right) \\ \text{Sujeito a} \begin{cases} L^T A \leq V_0 \Leftrightarrow \left(A_1 + A_2 + \sqrt{2} A_3 - \frac{V_0}{L} \right) \leq 0 \\ A_1 \geq 0 \\ A_2 \geq 0 \\ A_3 \geq 0 \end{cases} \end{cases}$$

Como se pode verificar, neste problema existe maior número de variáveis que equações. De acordo com Christensen [9] neste caso é possível resolver o problema recorrendo às condições KKT. Para validar estas condições define-se inicialmente o Lagrangeano:

$$\mathcal{L} = C_1 \left(\frac{1}{A_1} + \frac{1}{A_2} + \frac{2\sqrt{2}}{A_3} \right) + \lambda_1 \left(A_1 + A_2 + \sqrt{2} A_3 - \frac{V_0}{L} \right) \tag{2.38}$$

Como as áreas são todas maiores que zero ($A_i > 0$) e considerando que $\lambda_1 \neq 0$ e que a equação correspondente à restrição não está automaticamente verificada para todos os pontos do domínio, as seguintes condições devem de ser verificadas:

$$\frac{\partial \mathcal{L}(x, \lambda)}{\partial A_i} = 0 \quad e \quad \lambda_i g_i(A) = 0 \tag{2.39}$$

Pela primeira condição, relativa ao estudo da estacionariedade do lagrangeano tem-se:

$$C_1 \begin{bmatrix} -\frac{1}{A_1^2} \\ -\frac{1}{A_2^2} \\ -\frac{2\sqrt{2}}{A_3^2} \end{bmatrix} + \lambda_1 \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.40}$$

Resolvendo o sistema conclui-se que:

$$\begin{cases} \lambda_1 = C_1 / A_1^2 \\ \lambda_1 = C_1 / A_2^2 \\ \lambda_1 = 2C_1 / A_3^2 \end{cases} \tag{2.41}$$

Igualando as soluções de λ_1 , $C_1/A_1^2 = C_1/A_2^2$ e $C_1/A_1^2 = 2C_1/A_3^2$ obtêm-se as seguintes proporcionalidades entre as áreas:

$$\begin{cases} A_1 = A_2 \\ A_3 = \sqrt{2} A_1 \end{cases} \tag{2.42}$$

A segunda condição de KKT já referida em 2.38 será então:

$$\lambda_1 \left(A_1 + A_2 + \sqrt{2}A_3 - \frac{V_0}{L} \right) = 0 \tag{2.43}$$

Os valores óptimos das variáveis podem ser calculados substituindo as relações referidas em (2.41) na expressão anterior. Atendendo que $\lambda_1 \neq 0$ conclui-se que:

$$\left(2A_2 + 2A_2 - \frac{V_0}{L} \right) = 0 \Rightarrow A_2^* = \frac{V_0}{4L} \tag{2.44}$$

$$\begin{cases} A_1^* = A_2^* = \frac{V_0}{4L} \\ \\ A_3^* = \frac{\sqrt{2}V_0}{4L} \end{cases} \tag{2.45}$$

$$Minf(A_1, A_2.A_3) = C_1 \left(\frac{1}{A_1} + \frac{1}{A_2} + \frac{2\sqrt{2}}{A_3} \right) = \frac{16LC_1}{V_0} \tag{2.46}$$

Importa referir que o número de restrições activas é inferior ao número de variáveis pelo que, o ponto obtido como sendo o mínimo, poderá não ser um mínimo global mas sim local [9]. Assim, torna-se necessário verificar a monotonia da matriz Hessiana para saber se efectivamente o valor obtido se trata do máximo global do problema:

$$\begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial A_1^2} & \frac{\partial^2 \mathcal{L}}{\partial A_1 \partial A_2} & \frac{\partial^2 \mathcal{L}}{\partial A_1 \partial A_3} \\ \\ \frac{\partial^2 \mathcal{L}}{\partial A_1 \partial A_2} & \frac{\partial^2 \mathcal{L}}{\partial A_2^2} & \frac{\partial^2 \mathcal{L}}{\partial A_2 \partial A_3} \\ \\ \frac{\partial^2 \mathcal{L}}{\partial A_1 \partial A_3} & \frac{\partial^2 \mathcal{L}}{\partial A_2 \partial A_3} & \frac{\partial^2 \mathcal{L}}{\partial A_3^2} \end{bmatrix} = \begin{bmatrix} \frac{2}{A_1^3} & 0 & 0 \\ 0 & \frac{2}{A_2^3} & 0 \\ 0 & 0 & \frac{4\sqrt{2}}{A_3^3} \end{bmatrix} \tag{2.47}$$

Como se pode constatar, a matriz Hessiana é composta por elementos nulos ou positivos e definida em todos os pontos. Logo conclui-se, efectivamente, que a solução obtida anteriormente é o mínimo global e portanto a solução óptima do problema.

Assumindo os seguinte valores dos parâmetros associados á estrutura:

$$P = 500KN \qquad E = 30Gpa$$
$$L = 1m \qquad V_0 = 1m^3$$

resulta que o valor de $C_1 = 0.0083KNm^3$.

Resolvendo analiticamente e, como anteriormente foi elaborado, procedeu-se à realização da tabela com os resultados e do gráfico correspondente à variação do erro relativo (2.32) que podem ser vistos na Tabela 2.2 e na Figura 2.13.

Através do gráfico pode-se verificar que, embora nas primeiras iterações tenham sido obtidos valores de erro bastantes elevados, ao fim de algumas iterações os algoritmos convergem rapidamente para a solução exacta. Inicialmente o

Solução analítica (kJ)	NLPQL		MATLAB	
	Nº iterações	Trabalho das forças (kJ)	Nº iterações	Trabalho das forças (kJ)
0.1328	13	0.1328	17	0.1328

Tabela 2.2: Minimização do trabalho das forças aplicadas - Resultados obtidos.

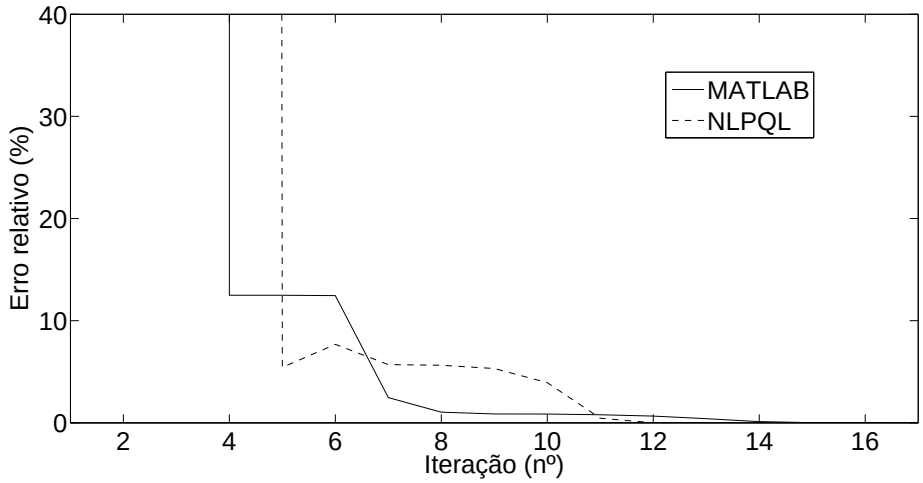


Figura 2.13: Evolução do erro ao longo do processo de optimização da rigidez da estrutura.

algoritmo NLPQL afasta-se bastante do resultado analítico, até que após a 5ª iteração o algoritmo atinge erros a rondar os 5 %. A partir daí o NLPQL vai convergindo lentamente para a solução exacta. Em relação ao algoritmo utilizado no MALTAB, o mesmo se verifica. Inicialmente também se verificam valores de erro elevados mas ao fim da 4ª iteração converge e atinge erros de 10 %. Se até à 7ª iteração o erro relativo é mais baixo para o algoritmo NLPQL, a partir daí tal não se verifica. Por volta da 11ª iteração, quando o erro relativo é praticamente nulo, o NLPQL volta a ter um valor do erro mais baixo. Pode-se assim concluir que mais uma vez o algoritmo NLPQL procede à optimização de problemas de forma bastante satisfatória. Os códigos utilizados nas subrotinas NLFUNC, NLGRAD e no MAIN PROGRAM podem ser consultados no **apêndice B**.

2.5 Conclusão

No final da resolução dos exercícios teste em que pretendia verificar o desempenho do algoritmo NLPQL é possível concluir que este algoritmo procede efectivamente à resolução deste tipo de problemas. Fá-lo em menos iterações que o algoritmo pré-definido no software MATLAB que, por estar inserido neste software, pressupõe-se que seja uma boa ferramenta de cálculo. Devido aos bons resultados obtidos e à sua rapidez de cálculo, este será este algoritmo o utilizado

neste trabalho.

Capítulo 3

Elementos Finitos Híbridos-Trefftz de deslocamento

3.1 Introdução

A formulação de elementos finitos híbridos-Trefftz tem vindo a procurar superar algumas limitações provenientes da forma convencional dos elementos finitos. Baseada no método de Trefftz apresentado por Erich Trefftz em 1926, esta metodologia é aplicada para a obtenção da solução de problemas estáticos e dinâmicos. Para tal, recorre a uma aproximação que satisfaça localmente todas as condições do domínio.

A primeira referência bibliográfica de que há registo, foi apresentada por Jirousek e remonta a 1978. Nela constava uma base para a implementação de elementos finitos de elevada dimensão que respeitavam todas as equações do domínio [20]. Ao longo destes anos esta temática tem sido alvo de interesse por parte de investigadores, entre os quais o grupo de investigação de análise estrutural da Universidade Técnica de Lisboa.

Este capítulo tem como objectivo a familiarização com a formulação usada nos elementos finitos híbridos-Trefftz, sem os descrever exaustivamente. Uma análise mais profunda sobre elementos finitos híbridos-Trefftz pode ser consultada em [10].

3.2 Equações fundamentais

Considere-se o corpo genérico de domínio V^e e de fronteira Γ^e conforme apresentado na Figura 3.1. Observando o corpo pode-se distinguir duas zonas distintas pertencendo à fronteira, consoante o tipo de ligação que o corpo tem com o exterior. Assim, a zona de fronteira onde são impostas restrições de deslocamento do corpo designa-se por fronteira cinemática (Γ_u^e) e a restante parte da fronteira define-se por fronteira estática (Γ_σ^e), na qual é conhecido o valor das forças exteriores aplicadas. Tratam-se, portanto de sub-regiões complementares

tendo obrigatoriamente de ser verificadas as condições ($\Gamma_u^e \bigcup \Gamma_\sigma^e = \Gamma^e$ e $\Gamma_u^e \bigcap \Gamma_\sigma^e = 0$).

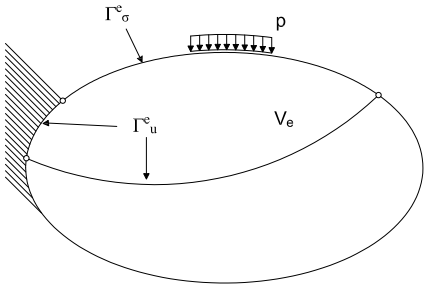


Figura 3.1: Corpo genérico de domínio V_e.

As relações fundamentais referentes à compatibilidade, ao comportamento mecânico dos materiais ou ao equilíbrio podem ser expressas por:

$$\varepsilon = \mathfrak{D}^* u \quad em \quad V^e \qquad (3.1)$$

$$\sigma = k\varepsilon \quad em \quad V^e \qquad (3.2)$$

$$\mathfrak{D}\sigma + b = 0 \quad em \quad V^e \qquad (3.3)$$

$$t = t_\Gamma \quad em \quad \Gamma_\sigma^e \qquad (3.4)$$

$$N\sigma = t \quad em \quad \Gamma^e \qquad (3.5)$$

$$u = u_\Gamma \quad em \quad \Gamma_u^e \qquad (3.6)$$

Onde nas equações de equilíbrio (3.5) e (3.3) σ corresponde ao vector das tensões, b e t correspondem aos vectores associadas às forças de massas e à aproximação das tracções, respectivamente. A matriz D representa a matriz operadora diferencial.

Na expressão referente à relação constitutiva (3.1) foi ignorada a parcela relativa à tensão residual. O vector ε corresponde às extensões e k à matriz de elasticidade. Na expressão (3.2) é referenciado o vector u que corresponde à aproximação do deslocamentos do corpo. As expressões (3.4) e (3.5) traduzem as condições de fronteira associadas ao corpo.

3.3 Aproximação dos deslocamentos

O vector de aproximação dos deslocamentos apresentado em (3.1), para o domínio do elemento finito, pode ser feito através da seguinte expressão:

$$u = U_1 q_1 + U_2 q_2 + u_p \qquad em \quad V^e \qquad (3.7)$$

Onde q_1 representa o peso da função de aproximação das deformações que é reunida em U_1. Da mesma maneira o vector q_2 representa a ponderação do deslocamento de corpo rígido U_2. Esta matriz é expressa num sistema de

coordenadas cartesianas com o centro no baricentro de cada elemento finito, e toma a seguinte forma:

$$U_2 = \begin{bmatrix} 1 & 0 & y \\ 0 & 1 & x \end{bmatrix} \tag{3.8}$$

Quanto à matriz U_1 pressupõe-se que esta seja a solução da equação homogénea de Navier e o vector u_p a solução particular. A aproximação das funções dos deslocamentos U_1 pode ser obtida recorrendo à derivação de funções potenciais, que são solução das equações diferenciais referentes à equação de equilíbrio. Segundo Cismasiu [10] existe outra opção para a definição da matriz U_1 que implica a utilização de equações potenciais bi-harmónicas governativas do estado de tensão, com recurso às coordenadas polares para problemas homogéneos, isotrópicos e linearmente elásticos. Este tipo de solução foi apresentado por G.B. Airy em 1862, por vezes é designado por *Airy stress function* e é expressa por:

$$\left(\frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2}\right)\left(\frac{\partial^2 \phi}{\partial r^2} + \frac{1}{r}\frac{\partial \phi}{\partial r} + \frac{1}{r^2}\frac{\partial^2 \phi}{\partial \theta^2}\right) = 0 \tag{3.9}$$

onde ϕ representa a função das tensões. A solução completa da equação anterior resulta numa combinação de funções potenciais,

$$\varphi = \left\{ \; r^n T_n \;\; r^n T_{n-2} \;\; lnr \;\; r^2 lnr \;\; \theta \;\; r^2 \theta \; \right\} \tag{3.10}$$

onde T_n representa uma das funções trigonométricas $sin(n\theta)$ ou $cos(n\theta)$ e $-\infty < n < \infty$.

Como referido u_p corresponde à solução particular da aproximação dos deslocamentos, que resulta de:

$$u_p = u_b + u_\pi \tag{3.11}$$

em que a parcela u_b corresponde a efeitos associados a forças de massa ou deformações residuais, e u_π que permite modelar efeitos devido a cargas pontuais ou mesmo deslocamentos impostos. Quando, sobre uma superfície livre, é aplicada uma carga pontual, é a parcela u_π a responsável pela modelação que é feita através da solução de Boussinessq para um domínio semi infinito [10].

3.4 Aproximação das tracções

O vector referente à aproximação das tracções (t) associadas às fronteiras cinemáticas (Γ_u^e) pode ser definido através da seguinte formulação:

$$t = Tp \qquad\qquad em \quad \Gamma_u^e \tag{3.12}$$

Na expressão acima, T representa a matriz representativa das funções de aproximação das tracções e o vector p reúne os pesos atribuídos à aproximação das tracções. A aproximação torna-se mais eficiente quando na matriz T são

utilizados polinómios de Chebyshev que são linearmente independentes e podem ser agrupados matricialmente sobe a forma:

$$T_n = I_2 cos(n cos^{-1}\xi) \tag{3.13}$$

onde a I_2 corresponde à matriz identidade de tamanho 2x2. Esta formulação é feita recorrendo a um novo sistema de eixos localizados no centro de cada fronteira e com intervalo $-1 \leq \xi \leq +1$. Assim, garante-se que a precisão proveniente do processo de cálculo é superior do que quando são utilizados polinómios normais [10].

3.5 Sistema Governativo

Conforme apresentado no trabalho desenvolvido por Cismasiu [10], da combinação das equações referentes ao equilíbrio, da compatibilidade e elasticidade resulta um conjunto de equações que organizadas matricialmente originam um sistema governativo que rege a formulação híbrida-Treffz para problemas bi-dimensionais e elasticamente perfeitos:

Equação de equilíbrio	Equação de compatibilidade	Equação de elasticidade
$\begin{Bmatrix} X_i \\ 0 \end{Bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} p + \begin{Bmatrix} X_{01} \\ X_{02} \end{Bmatrix}$	$v_\Gamma - v_p = \begin{bmatrix} B_1^T & B_2^T \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix}$	$X_1 = K q_1$

Tabela 3.1: Equações que regem o sistema governativo.

$$\begin{bmatrix} K & 0 & -B_1 \\ 0 & 0 & -B_2 \\ -B_1^T & -B_2^T & 0 \end{bmatrix} = \begin{Bmatrix} q_1 \\ q_2 \\ p \end{Bmatrix} = \begin{Bmatrix} X_{01} \\ X_{02} \\ v_p - v_\Gamma \end{Bmatrix} \tag{3.14}$$

Onde:

$$K = \int U_1^T N\ k\ \mathfrak{D}^*\ U1\ \mathrm{d}\Gamma^e \qquad X_{01} = \int U_1^T\ t_\Gamma\ \mathrm{d}\Gamma_\sigma^e$$

$$B_1 = \int U_1^T\ T\ \mathrm{d}\Gamma_u^e \qquad X_{02} = \int U_2^T t_\Gamma\ \mathrm{d}\Gamma_\sigma^e$$

$$B_2 = \int U_2^T\ T\ \mathrm{d}\Gamma_u^e \qquad v_\Gamma = \int T^T u_\Gamma\ \mathrm{d}\Gamma_u^e$$

$$X_p = \int U_1^T\ N\ \sigma_p\ \mathrm{d}\Gamma^e \qquad v_p = \int T^T u_p\ \mathrm{d}\Gamma^e$$

3.6 Indeterminação estática e cinemática

A indeterminação estática e cinemática representa o número de graus de liberdade das equações de equilíbrio e de compatibilidade, respectivamente. Consideremos n_1 e n_2 como o tamanho do vector que guarda os pesos atribuídos à aproximação dos deslocamentos q_1 e q_2. Assim, $n_u = n_1 + n_2$ define o número de graus de liberdade cinemáticos. Do mesmo modo podemos definir o número de graus de liberdade estáticos (n_p) como sendo o tamanho do vector p. Assim o número de indeterminação estática (α) e cinemática (β) são dados por:

$$\alpha = n_p - n_2 \tag{3.15}$$

$$\beta = n_1 + n_2 - n_p \tag{3.16}$$

Estas duas expressões mostram que os elementos são estaticamente indeterminàveis quando $n_p >> n2$, que toma valor de 3, por se tratar das componentes correspondentes ao movimento de corpo rígido em plano. A fim de evitar problemas de consistência nas equações compatibilidade, Cismasiu [10] propõe que o valor de β deve tomar valor não nulo e que deverá tomar o menor valor possível. Assim, para assegurar uma forte ligação entre os elementos em termos de tensões, o valor de indeterminação cinemática deverá obedecer à seguinte regra expressa em (3.17), em que d_u representa o grau de aproximação da função de deslocamentos, d_t o grau da função de aproximação das tensões e n_t número de fronteiras onde se aproximam as tensões (Γ_u^e).

$$\beta = 4d_u + 6 - 2n_t(d_t + 1) \geq 0 \tag{3.17}$$

3.7 Programa de cálculo HTD

Para a resolução do método dos elementos finitos híbridos-Trefftz foi utilizado um programa de cálculo desenvolvido e disponibilizado pelo Professor Corneliu Cismasiu. O programa em causa, à semelhança do NLPQL, foi desenvolvido em linguagem de programação Fortran.

Um esquema representativo do funcionamento do programa é apresentado na Figura 3.2. Como se pode verificar logo no início do programa de cálculo, é pedido ao utilizador que forneça um ficheiro de entrada. Este ficheiro deverá conter toda a informação referente ao problema, para que através de um único ficheiro o programa de cálculo tenha todos os dados necessários e proceda à resolução. Portanto, antes da execução do programa o utilizador é obrigado a desenvolver o ficheiro que contém toda a informação referente ao problema. Segundo um critério imposto, o utilizador fornece dados como a geometria da estrutura, número de elementos, características do material e o carregamento a aplicar na estrutura. Neste ficheiro também deverá estar indicado qual o grau do polinómio de aproximação das tensões (d_t) e dos deslocamentos (d_u) a aplicar na resolução do problema. Um exemplo deste ficheiro de entrada e, que foi utilizado

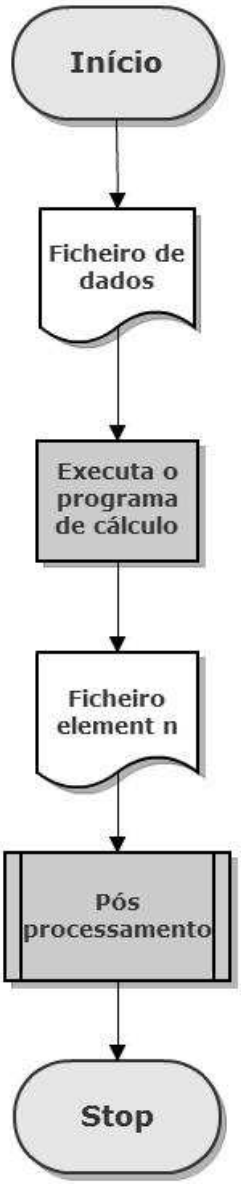


Figura 3.2: Fluxograma representativo do funcionamento do programa de cálculo HTD.

no primeiro dos exercícios resolvidos, é apresentado no **apêndice D**.

Após a criação do ficheiro de dados, o utilizador poderá então executar o programa de cálculo automático. Deste processo resulta um conjunto de ficheiros ao qual na Figura 3.2 se deram o nome de $element\ n$. Nestes ficheiros encontram-se registadas algumas informações referentes aos elementos finitos considerados. Se, por exemplo, se dividir se uma estrutura em dois elementos finitos serão criados dois ficheiros cada um deles associados a um elemento. Neste ficheiro encontram-se registadas várias informações como, por exemplo, a área de cada um dos elementos.

A informação do estado de tensão da peça assim como os seus deslocamentos podem ser obtidos através a execução do pós processamento. O funcionamento do pós-processamento é explicado na Figura 3.3.

Depois de executar o programa de cálculo é possível ao utilizador seleccionar que informação deseja obter.

Seguindo a referida figura, a primeira opção prende-se com a alteração do número de divisões. Esta opção permite obter melhores representações gráficas dos problemas. Se, por exemplo, considerar-se uma fronteira de um elemento finito, ao escolher um número mais elevado de divisões, possibilita-se que sejam calculados os esforços em mais pontos dessa fronteira, aumentando assim o número de pontos sobre os quais se obtém a informação do estado de tensão. Se ampliar este conceito não só para uma fronteira mas sim para mais linhas do elemento, proporciona-se uma melhor visualização dos resultados pois a informação recolhida para traçar o gráfico é maior.

Escolhido o número de divisões, é perguntado ao utilizador se pretende obter o estado de tensão em alguma das fronteiras dos elementos finitos. Em caso de uma resposta positiva, o utilizador terá de introduzir o número de fronteiras, identifica-las e mais uma vez escolher o número de divisões que deseja. No final deste processo é criado um ficheiro chamado $Stress\ Table$ onde fica registada toda a informação solicitada. Em caso de não pretender nenhuma destas informações passará para a próxima questão.

Posteriormente é perguntado se deseja obter o estado de tensão em um outro qualquer ponto do domínio da estrutura. Caso o pretenda, o utilizador mais uma vez indica o número de pontos e identifica-os. No caso do ficheiro $Stress\ Table$ já existir, é actualizado com a nova informação, caso contrário é nesta altura criado. No caso de não querer obter as tensões num ponto interior passará automaticamente para a pergunta referente aos deslocamentos.

O utilizador, depois de selecionar as informação que deseja relativamente às tensões na estrutura, pode escolher obter os deslocamentos nas fronteiras dos elementos finitos. Para tal terá de informar o número de fronteiras e identificá-las.

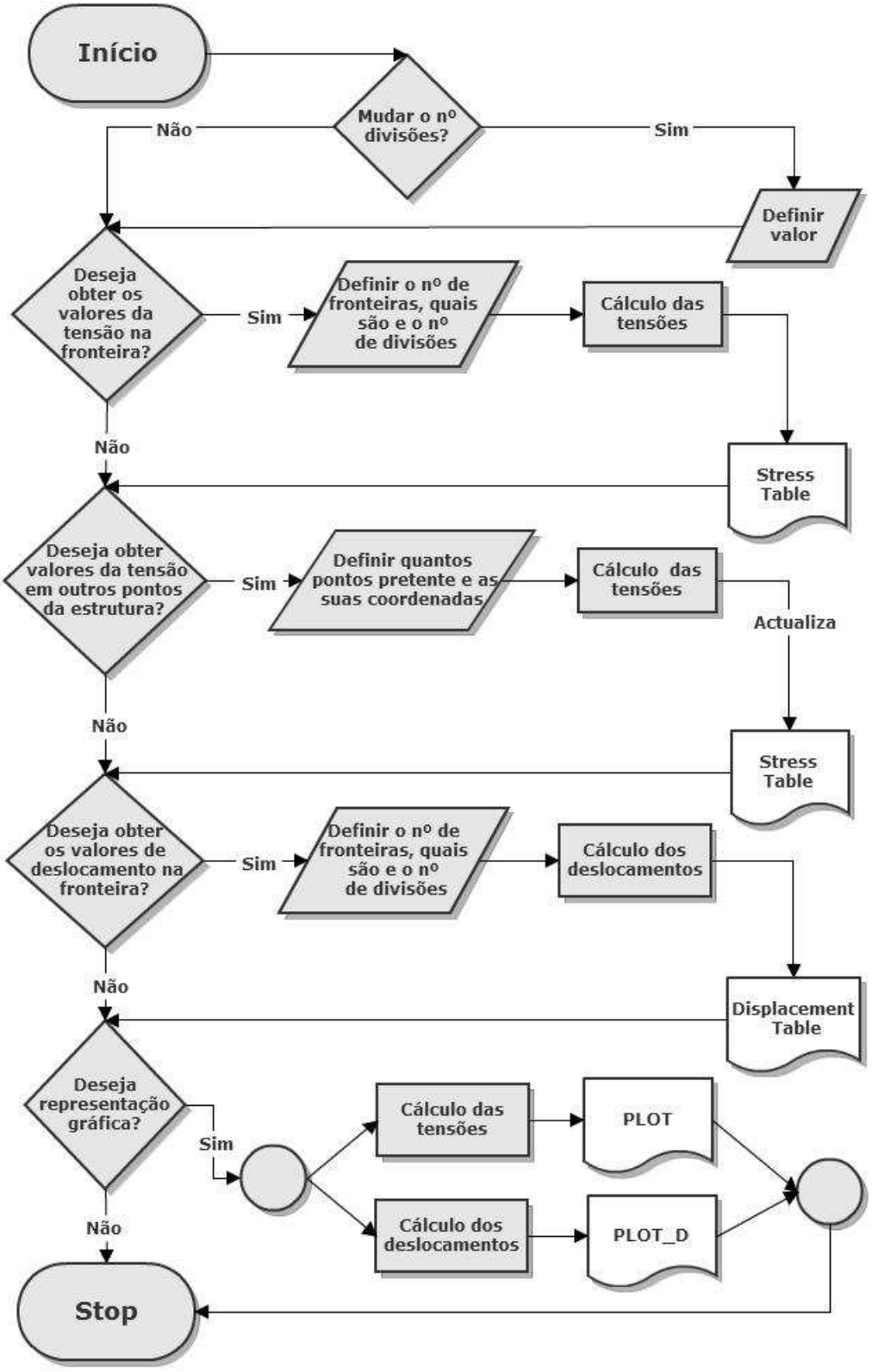


Figura 3.3: Fluxograma representativo do pós processamento do programa HTD.

Neste caso também lhe é perguntado quantas divisões quererá efectuar. No final é criado um ficheiro com o nome $Displacement\ Table$ onde fica armazenada toda esta informação.

Por fim é perguntado ao utilizador se deseja obter representação gráfica dos resultados. Em caso de o desejar, o programa fornece dois ficheiros. O primeiro, com o nome $Plot.vtk$ recolhe a informação do estado de tensão num número elevado de pontos do domínio da estrutura. O segundo ficheiro guarda a informação dos deslocamentos na fronteira de cada um dos elementos e tem o nome de $Plot\_D.vtk$. Para a obtenção de boas representações gráficas é aconselhável que se considere um número de divisões elevada. Com os ficheiros criados e com recurso ao software Paraview consegue-se visualizar os campos de tensões e de deslocamentos das estruturas. O Paraview é um software que permite de forma rápida a visualização de um grande conjunto de dados, quer sejam dados referente a problemas em 2D ou em 3D [2].

3.8 Exemplos de Aplicação

Neste subcapítulo pretende-se averiguar o desempenho dos elementos finitos híbridos-Trefftz face aos resultados obtidos através de dois softwares, o SAP2000 [11] e o ANSYS [1], que utilizam no seu processo de cálculo elementos finitos convencionais. Para tal, foram analisadas algumas estruturas simples através dos três programas de cálculo e comparados os resultados.

3.8.1 Consola curta

Com primeiro exemplo pretende-se analisar a evolução das tensões num ponto, fazendo variar o número de graus de liberdade. Considerou-se uma consola sujeita a uma força de tracção conforme apresentado na Figura 3.4. O ponto a ser analisado é o ponto A com coordenadas (0.25;0.25). Considera-se que o material tem o valor de 1kPa para o módulo de elasticidade e 0.3 para o coeficiente de Poisson.

Na modelação numérica recorrendo ao programa HTD, foi considerado apenas um elemento finito e a aproximação das tracções foi feita através de um polinómio de grau 7. A variação dos graus de liberdade foi feita através do aumento do polinómio de aproximação dos deslocamentos.

A solução do problema através do software SAP2000 implicou a utilização de elementos finitos quadriculares de quatro nós. O aumento dos graus de liberdade fez-se aumentando o número de elementos na estrutura que, consequentemente, faz aumentar o número de nós e por sua vez os graus de liberdade.

A modulação efectuada através do software ANSYS é bastante semelhante à descrita anteriormente. Neste caso, o tipo de elementos finitos utilizado foi o PLANE82. Este tipo de elementos está disponível neste software [1]

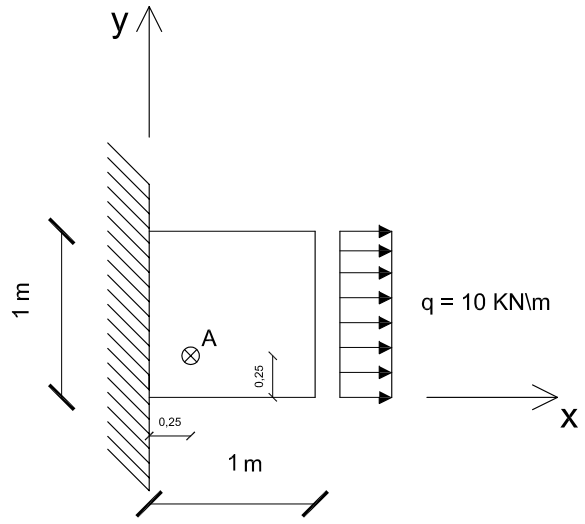


Figura 3.4: Consola sujeita a um carregamento.

e caracteriza-se por serem quadriculares e conterem oito nós, quatro deles localizados nos vértices e os restantes quatro a meio de cada uma das arestas do elemento.

Os resultados obtidos são apresentados na Tabela 3.2 e nos gráficos apresentados na Figura 3.5.

	HTD			SAP2000			ANSYS		
N	70	90	118	50	162	338	40	130	450
d_u	13	18	25	-	-	-	-	-	-
σ_{xx}	10.14	10.13	10.13	9.99	10.10	10.14	9.99	10.14	10.14
σ_{yy}	0.67	0.66	0.66	0.46	0.57	0.61	1.18	0.75	0.63
τ_{xy}	0.37	0.32	0.33	0.45	0.46	0.44	0.44	0.52	0.33

Tabela 3.2: Resultados obtidos para a tensão no ponto A (em kPa).

A partir dos resultados apresentados na tabela pode-se verificar que o método dos elementos finitos híbridos-Trefftz necessita de um número de graus de liberdade mais reduzido. Com 90 ou 118 graus de liberdade esta formulação atinge resultados para os quais o ANSYS ou SAP2000 necessitam cerca de 450 e 338 graus de liberdade, respectivamente.

Através dos gráficos pode-se constatar que o HTD converge mais rapidamente que os restantes dois programas. No gráfico referente à tensão normal ao eixo x, a convergência para o HTD dá-se para 70 graus de liberdade face aos outros métodos que necessitam de um número muito superior. As mesmas conclusões se podem tirar para os restantes gráficos (σ_{yy} e τ_{xy}).

Na realidade, os resultados obtidos com recurso ao ANSYS e ao SAP2000

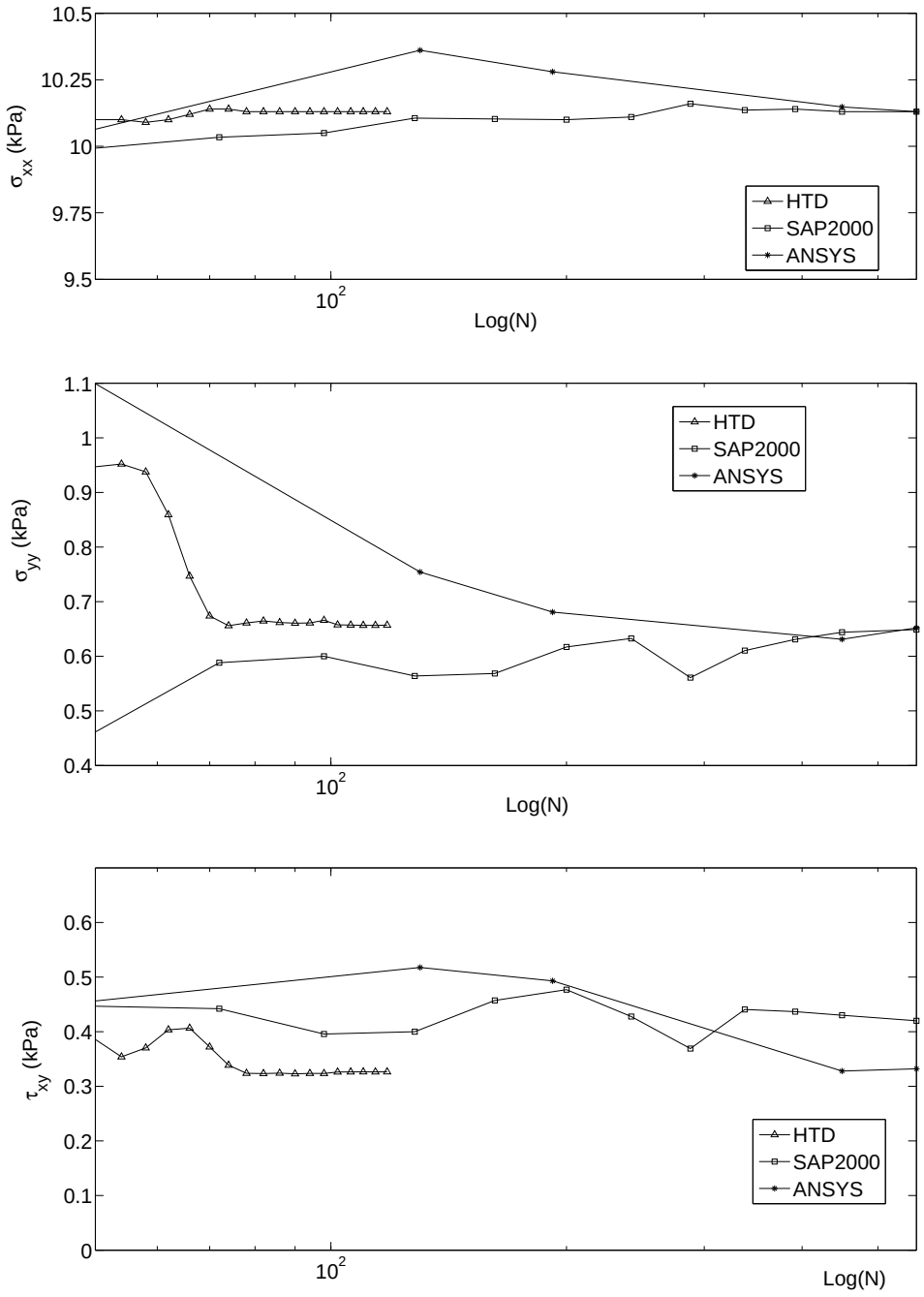


Figura 3.5: Evolução das tensões - Resultados obtidos para os diversos programas de cálculo.

são grosseiros. A divisão da consola foi feita para que fossem obtidos quadrados do mesmo tamanho. Por exemplo, optou-se por dividir as arestas da consola em 8 vezes resultando, assim 64 elementos finitos todos com mesmo tamanho. Para este caso não seria a melhor opção pois é sabido que nos vértices correspondentes ao encastramento existirão concentrações de tensões. Dever-se-ia, portanto, efectuar um refinamento da malha nessas zonas. Mas por outro lado, evidencia um dos pontos positivos do HTD que é o facto de não precisar de um refinamento da malha e mesmo assim obter-se bons resultados.

3.8.2 Exemplo de uma placa simplesmente apoiada

Com este problema pretende-se analisar a influência que a solução particular toma na aproximação dos deslocamentos, corresponde à parcela u_p. Assim, foi considerada uma placa simplesmente apoiada sobre a qual é aplicada um carga a meio vão conforme é apresentado na Figura 3.6. Foram realizados dois testes. No primeiro considerou-se que a parcela correspondente a solução particular tomaria valor nulo ($u_p = 0$). No segundo teste esta parcela corresponderia a solução de Boussinesq de forma a modelar os efeitos locais da aplicação da força concentrada a meio vão.

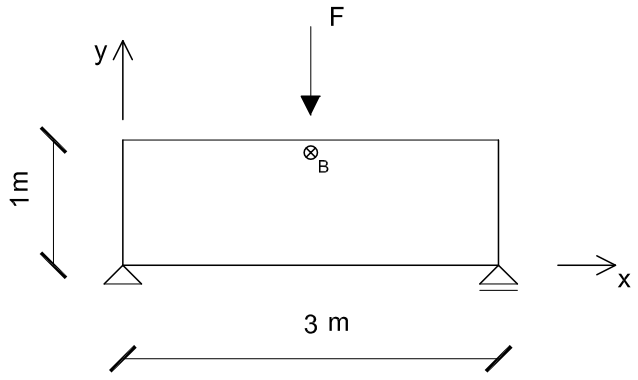


Figura 3.6: Placa bi-apoiada sujeita a uma força concentrada.

Neste exercício o módulo de elasticidade toma o valor de E=1 kPa, o coeficiente de Poisson toma o valor de $\upsilon = 0.3$ e a força aplicada toma o valor de 1 kN. Este problema foi mais uma vez modelado através dos softwares ANSYS e SAP2000 e comparados os seus resultados com o HTD. O ponto analisado tem como coordenadas (1.5,0.9).

Para os dois testes foram utilizados apenas um elemento finito híbrido-Trefftz. A variação dos graus de liberdade foi feita através do aumento do polinómio de aproximação dos deslocamentos. Neste caso, o polinómio de aproximação das tracções pode ser ignorado pois apenas se trabalhará com um elemento e as condições de apoio são pontuais.

Em relação aos softwares não foi alterado o tipo de modelação utilizado

no exercício anterior. Para o software SAP2000 foram utilizados elementos quadriculares de quatro nós e no ANSYS elementos quadriculares de oito nós (PLANE82).

	HTD				SAP2000		ANSYS	
	Com função particular		Sem função particular					
N	25	45	25	45	90	306	106	354
d_u	5	10	5	10	-	-	-	-
σ_{xx}	-2.23	-2.61	-2.82	-3.52	-3.31	-2.74	-3.19	-2.72
σ_{yy}	-6.53	-6.28	-1.13	-1.58	-2.92	-5.38	-2.14	-6.13
τ_{xy}	0	0	0	0	0	0	0	0

Tabela 3.3: Resultados obtidos para a tensão no ponto B (em kPa).

Os resultados obtidos estão organizados na Tabela 3.3. Com a parcela correspondente à solução particular inserida, na função de aproximação dos deslocamentos, consegue-se uma melhoria significativa para os valores da tensão. Mais uma vez o HTD alcança praticamente os resultados obtidos através dos dois softwares. Mas, para tal, necessita de menos graus de liberdade. Com apenas 45 graus de liberdade o HTD atinge os resultados obtidos através do ANSYS que utilizou 354 graus de liberdade e do SAP2000 que recorreu a 306. Mesmo sem recorrer à função de Boussinesq pode-se verificar que os valores da tensão no ponto não são muito diferentes. Os 45 graus de liberdade utilizados no HTD assemelham-se aos 90 e 106 graus utilizados no SAP e no ANSYS, respectivamente. De seguida pode-se verificar graficamente a influência da solução particular na aproximação dos deslocamentos.

Como se pode constatar, através da Figura 3.7, com utilização da solução de Boussinesq como parte integrante da aproximação dos deslocamentos, obteve-se uma melhoria na definição dos campos de tensão a que a peça está sujeita. Assim, concluí-se que o uso desta parcela trás benefícios quando são aplicadas cargas pontuais sobre as estruturas.

3.8.3 Exemplo de uma placa em forma de L

A ideia deste exercício prende-se com a sensibilidade que é necessário ter na escolha dos graus de liberdade a considerar na estrutura. A possibilidade de ser o utilizador a definir o grau do polinómio da função de aproximação dos deslocamentos (d_u) e das tracções (d_t), que estão relacionadas com o cálculo dos graus de liberdade, dá ao utilizador inexperiente uma grande liberdade fazendo com que não tire o melhor proveito desta formulação.

Assim, foi considerada a estrutura apresentada na Figura 3.8, com um módulo de elasticidade de 1kPa e um coeficiente de Poisson de 0.1. Com este estudo

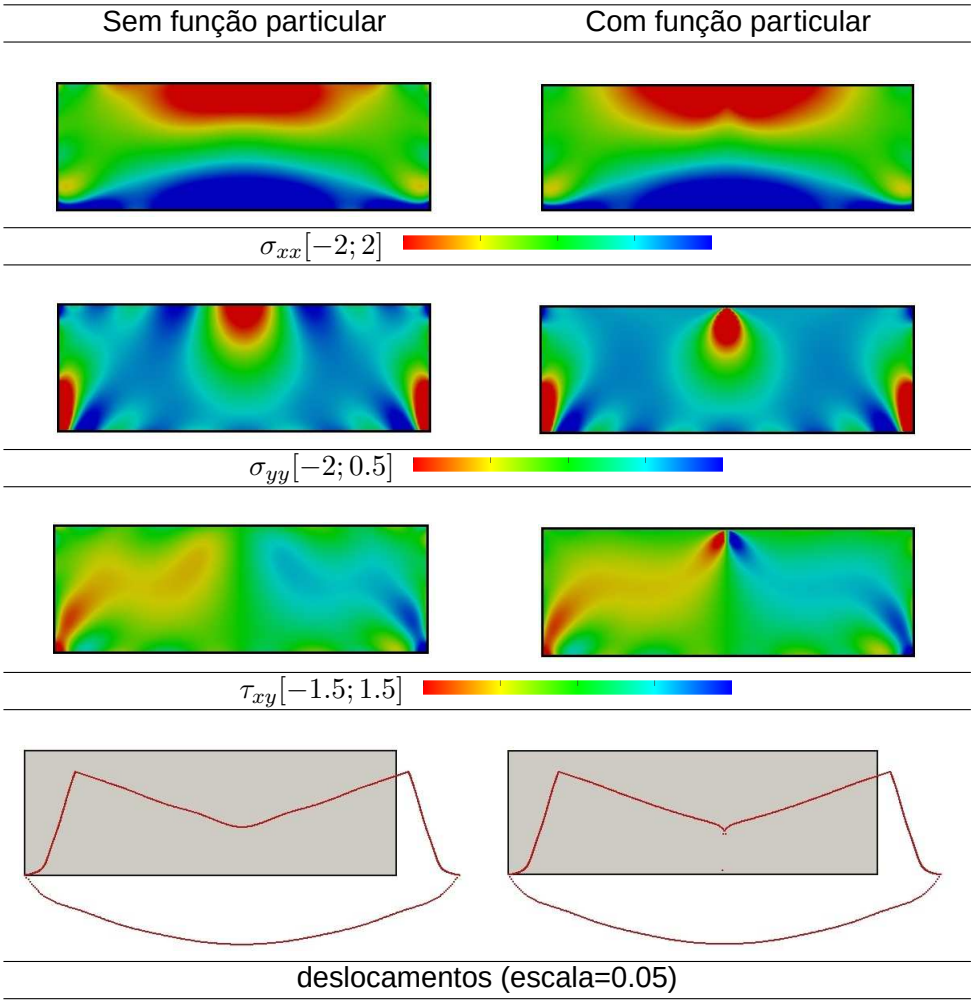


Figura 3.7: Representação gráfica da solução para a viga, utilizando apenas um único elemento finito HTD.

pretendeu-se calcular a energia de deformação, mantendo o grau da função de aproximação das tensões (d_t) e fazendo variar o grau do polinómio da função de aproximação dos deslocamentos (d_u). Foram então realizados 3 casos de estudo, um para $d_t = 3$, outro para $d_t = 5$ e por fim $d_t = 7$. Na aproximação das tracções foram utilizados polinómios de Chebyshev pois garantem a obtenção de melhores resultados. No final traçou-se um gráfico para cada um dos casos, onde se pretende verificar a evolução da energia de deformação para vários valores de d_u.

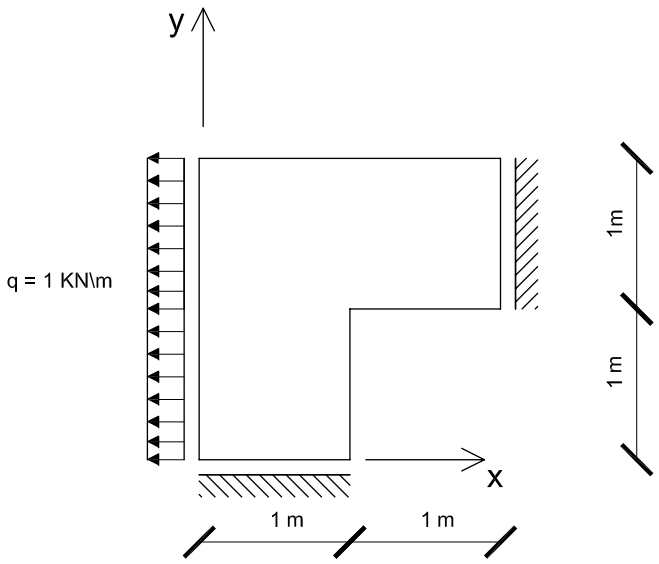


Figura 3.8: Placa em forma de L sujeita á tracção.

Na resolução deste problema apenas foi considerado um elemento finito. Devido à geometria da peça seria possível implementar uma aproximação do campo de tensões na zona do ponto singular (1,1), com a qual se obteria uma melhor aproximação das tensões e dos deslocamentos. Esta solução não foi utilizada para este estudo ficando apenas centrado na convergência da energia de deformação. Os resultados são apresentados na Figura 3.9.

Como se pode verificar, para cada um dos 3 testes, à medida que se aumenta o grau do polinómio a energia de deformação tende a convergir. Todavia, o aumento exagerado do grau do polinómio associado à aproximação dos deslocamentos poderá não trazer melhorias significaticas.

Assim, pode-se concluir que a escolha dos graus das funções de aproximação nesta formulação são vitais para uma boa solução do problema. Para utilizadores inexperientes este factor poderá ser preponderante pois nem sempre um polinómio de elevado grau poderá trazer melhorias à solução do problema. É de salientar que conforme aumenta o grau de aproximação dos deslocamentos, também aumenta o número de indeterminação cinemática (β) que se quer sempre positivo mas o mais baixo possível.

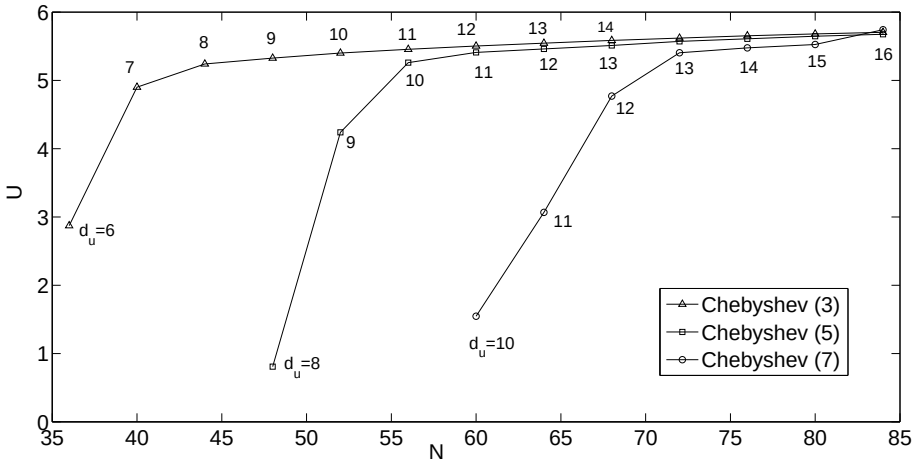


Figura 3.9: Resultados obtidos para a energia de deformação para os diversos cenários.

3.9 Conclusão

Depois da realização dos exercícios apresentados pode-se concluir que com o método dos elementos finitos híbridos-Tefftz permite obter resultados bastantes satisfatórios. Com um número de graus de liberdade mais baixo e com menos elementos finitos consegue-se atingir os mesmos resultados que utilizandao a formulação convencional dos elementos finitos. Por estes motivos, este método será utilizado ao longo deste trabalho.

Capítulo 4

Optimização de forma utilizando HTD e NLPQL

Neste capítulo pretende-se explicar como se incorporou o programa de cálculo automático HTD no NLPQL, permitindo assim a optimização de forma em estruturas. Recordando o que foi dito no subcapítulo 2.4, para proceder à resolução de problemas de optimização o utilizador terá de desenvolver duas subrotinas, a NLFUNC e a NLGRAD. Assim, e para cada uma das subrotinas foi desenvolvido um algoritmo que permite a optimização de problemas recorrendo a elementos finitos híbridos-Trefftz.

4.1 Algoritmo NLFUNC

Conforme foi analisado no subcapítulo 2.4, no final da execução desta subrotina, não interessava saber quais as expressões analíticas das funções do problema, mas sim o valor que estas tomavam. Esta observação é bastante importante pois o programa HTD no final da sua execução apenas informa os valores referentes à área, às tensões e aos deslocamentos. Pelo que, a cada iteração do algoritmo NLPQL pode-se executar o programa HTD e através do mesmo obter a informação necessária para definir as funções do problema.

O algoritmo a ser implementado na subrotina NLFUNC apresenta duas grandes metas. A primeira meta envolve a execução do programa HTD e o pós-processamento de forma automática. Na segunda meta procura-se recolher a informação necessária para definir as funções aos ficheiros criados no final da execução do programa HTD. Após este deste processo (e já com a informação necessária) procede-se à definição das funções do problema. Para atingir cada um dos dois objectivos foi necessário criar um conjunto de tarefas. Na Figura 4.1 é apresentado o conjunto de passos a seguir para a implementação da subrotina NLFUNC.

Relativamente à primeira meta, para executar o programa HTD (subcapítulo 3.7) é necessário criar um ficheiro de entrada. De notar que é neste ficheiro onde estão registados todos os parâmetros associados ao problema, inclusive as

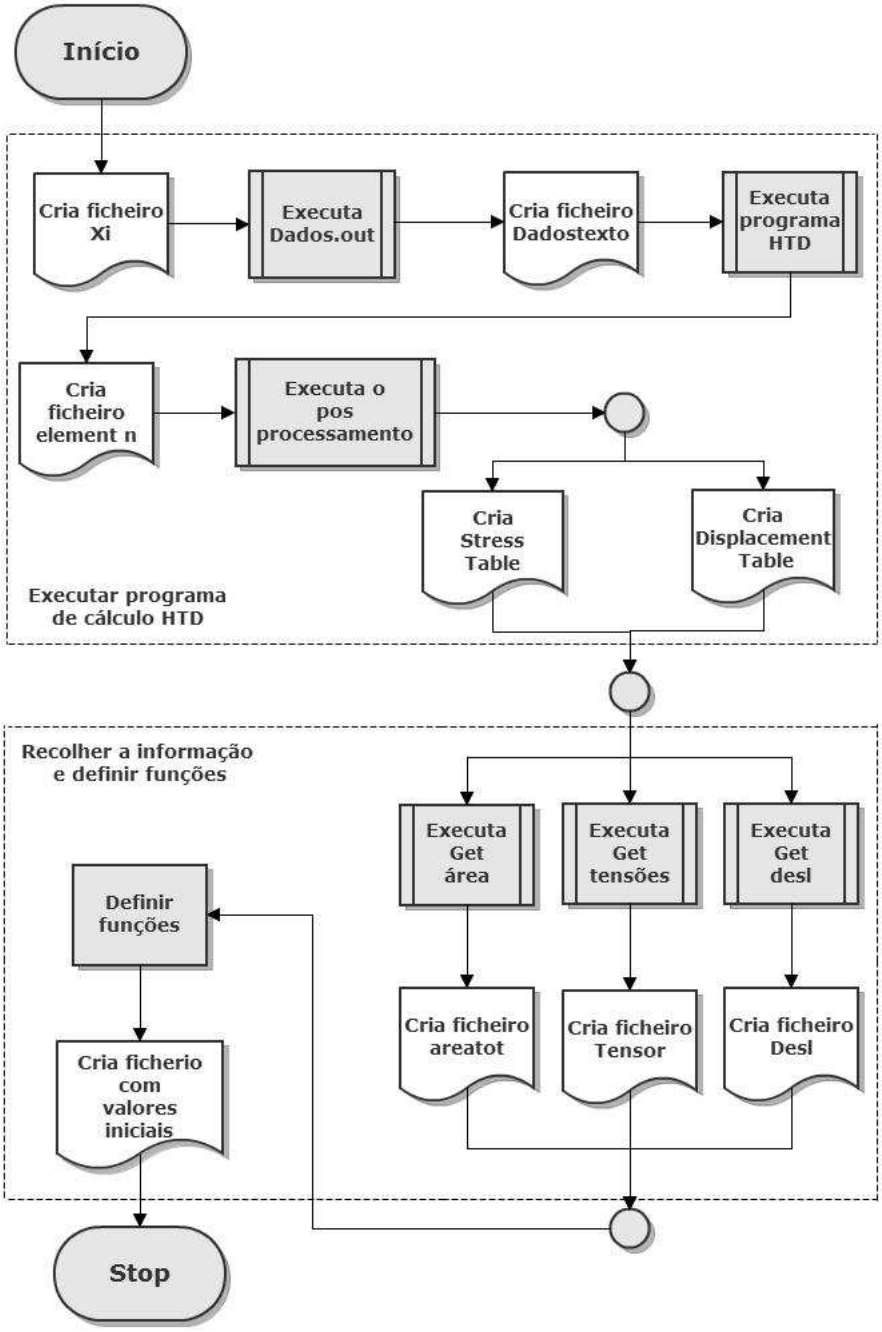


Figura 4.1: Algoritmo NLFUNC.

variáveis de projecto. É, portanto, necessário que, a cada iteração, os valores das variáveis sejam actualizados neste ficheiro. Para tal, a primeira tarefa a realizar é guardar os valores das variáveis num ficheiro ao qual se deu o nome Xi. De seguida, procede-se à execução de uma função chamada Dados.out. É através desta função que é obtido o ficheiro de entrada já com os valores das variáveis actualizados. Esta função começa por abrir o ficheiro Xi recolhe a informação que o mesmo contem e ao "re-escrever"' o ficheiro fá-lo com os valores das variáveis actualizados. No final é criado o ficheiro (ao qual se deu o nome de $Dadostexto$) a dar como entrada para a execução do programa HTD.

Segue-se a execução do programa de cálculo automático HTD e do respectivo pós-processamento. Deste processo, conforme foi visto no subcapítulo 3.7, resultam um conjunto de ficheiros que contêm a informação pedida pelo utilizador, atingindo-se, assim, a primeira meta.

Falta agora definir as funções que fazem parte do problema. Para tal, é necessário recolher a informação que se encontra nos ficheiros criados no final do processo de execução do programa HTD. Seria mais fácil se estes ficheiros apenas contivessem a informação estritamente pedida, contudo isso não acontece. Para proceder à recolha de dados estritamente necessários foram criadas três funções: $get\ area$, $get\ tensoes$ e $get\ desl$. Pretende-se que depois de executar estas funções, sejam criados três novos ficheiros que contenham unicamente a informação necessária para definir as funções. Assim a função $get\ area$ fica responsável por fornecer o valor da área da estrutura, a função $get\ tensoes$ por fornecer os valores das tensões nos pontos e a função $get\ desl$ por fornecer os valores dos deslocamentos. Uma explicação mais cuidadosa destas funções pode ser consultada mais a frente neste trabalho. Criados os três ficheiros pode-se definir as funções do problema, bastando para tal abrir os mesmos e proceder à sua leitura. Antes de finalizar o algoritmo é criado um ficheiro com todos os dados usados na definição das funções. Este ficheiro será depois utilizado na subrotina NLGRAD.

4.2 Algoritmo NLGRAD

O algoritmo a implementar na subrotina NLGRAD tem como objectivo o cálculo dos valores das derivadas parciais das funções integrantes do problema de optimização. Mais uma vez os resultados obtidos através do programa de cálculo HTD não são funções mais sim valores referentes ao estado de tensão ou de deslocamentos, pelo que tem de se recorrer a um processo que possibilite colmatar este aspecto. Neste sentido, o cálculo das derivadas parciais foi efectuado através do método das diferenças finitas, expresso através da equação (4.1).

$$\frac{\partial f}{\partial x_i}\left(x_1, ..., x_n\right) \cong \lim_{\delta \to 0} \frac{f\left(x_1, ..., x_i + \delta, ..., x_n\right) - f\left(x_1, ..., x_i, ..., x_n\right)}{\delta} \tag{4.1}$$

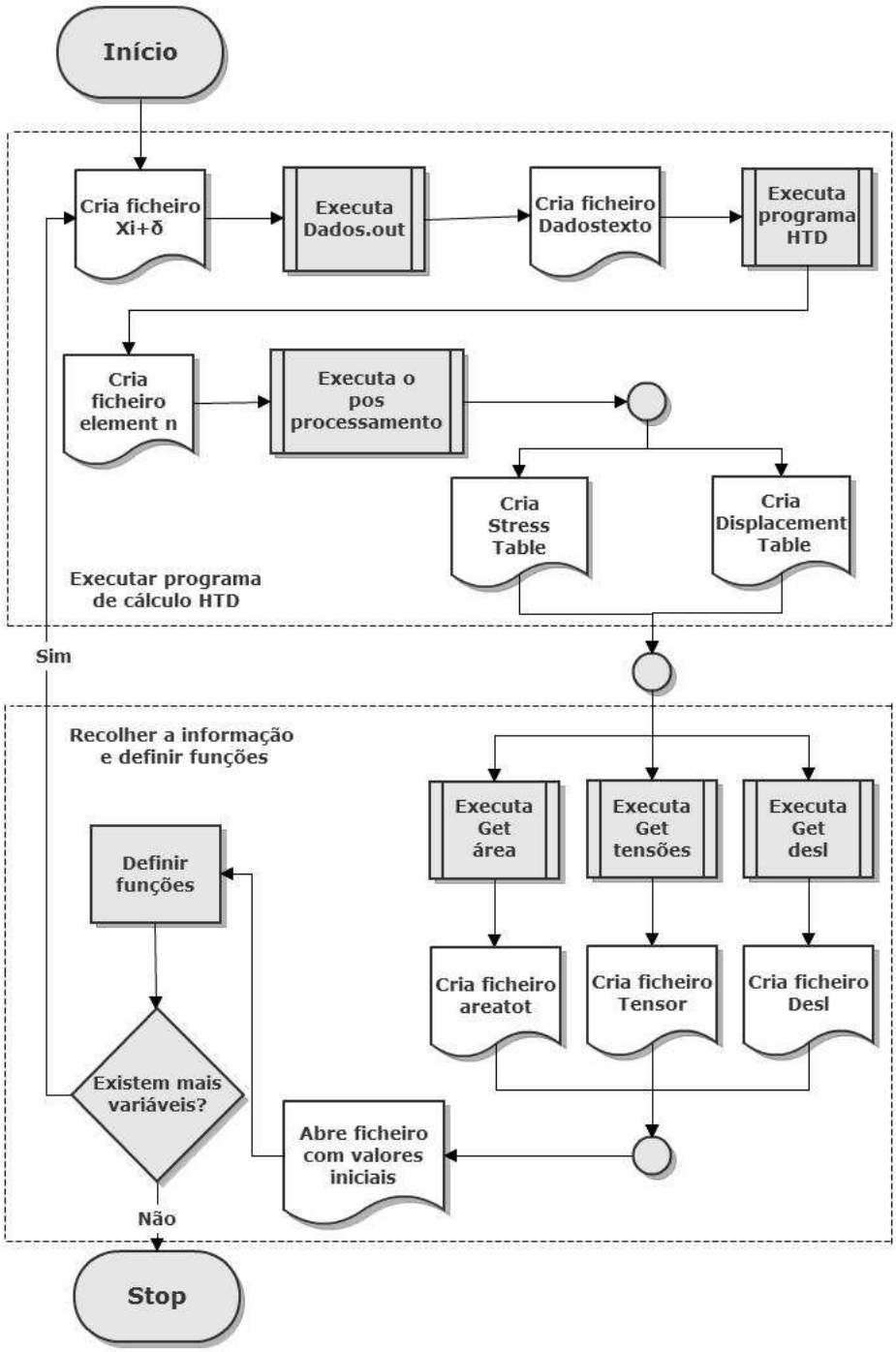


Figura 4.2: Algoritmo NLGRAD.

As duas metas a atingir para a definição das funções apresentadas no algoritmo NLFUNC também são válidas para o algoritmo NLGRAD, nomeadamente: executar o programa de cálculo HTD, recolher a informação e definir as derivadas parciais. Porém esta subrotina apresenta algumas diferenças.

Com o algoritmo NLFUNC bastava executar apenas uma vez o programa de cálculo HTD e conseguia-se definir todas as funções do problema. No algoritmo NLGRAD, no final de uma execução do programa HTD, apenas se conseguirá definir os valores das derivadas parciais associadas a uma variável. Terá assim que se executar o programa HTD tantas vezes quantas o número de variáveis que constituem o problema. Relembre-se que no final do algoritmo NLFUNC foi criado um ficheiro com os valores iniciais. A informação guardada nesse ficheiro é igual á parcela $f(x_1, ..., x_i, ..., x_n)$ da equação (4.1), sendo apenas necessário executar o programa HTD para obter a outra parcela $f(x_1, ..., x_i + \delta, ..., x_n)$.

O algoritmo implementado na subrotina NLGRAD é apresentado na Figura 4.2.

O algoritmo NLGRAD começa por criar o ficheiro com o valor das variáveis à excepção da variável Xi que toma o valor estabelecido mais o valor delta, definido pelo utilizador. Segue-se a execução da função $Dados.out$ do qual resulta o ficheiro $Dadostexto$ a dar de entrada no programa de cálculo HTD. Posteriormente executa-se o pós processamento do qual resultam os ficheiros que registam os valores da tensão, de deslocamento e o valor da área da estrutura.

Recorrendo às funções $get\ area$, $get\ tensoes$ e $get\ desl$ processa-se à recolha da informação necessária para definir as derivadas parciais associadas à variável Xi. Com os ficheiros obtidos anteriormente e com o ficheiro obtido no final da execução do algoritmo NLFUNC podem-se definir as derivadas parciais. Por fim, se este processo já se tiver realizado para todas as variáveis o algoritmo termina. Caso contrário todo o processo se repetirá para uma nova variável.

4.3 Função Dados.out

Pretende-se com a criação desta função que, sempre que necessário, "re-escrever"o ficheiro a dar de entrada para a execução do programa de cálculo HTD.

O algoritmo NLPQL é um processo iterativo e, como tal, para cada iteração serão obtidos novos valores para as variáveis. Assim, para cada nova iteração é preciso actualizar o ficheiro de entrada. Analise-se o exemplo apresentado na Figura 4.3.

As variáveis de projecto neste problema são as coordenadas do ponto 2. Assim, para a iteração i, os valores que estas variáveis tomam são de $x1 = 1$ e $x2 = 0$, sendo esta a informação colocada no ficheiro Xi. Ao executar a função $Dados.out$ é realizada a leitura do ficheiro Xi e no local referente às coordenadas

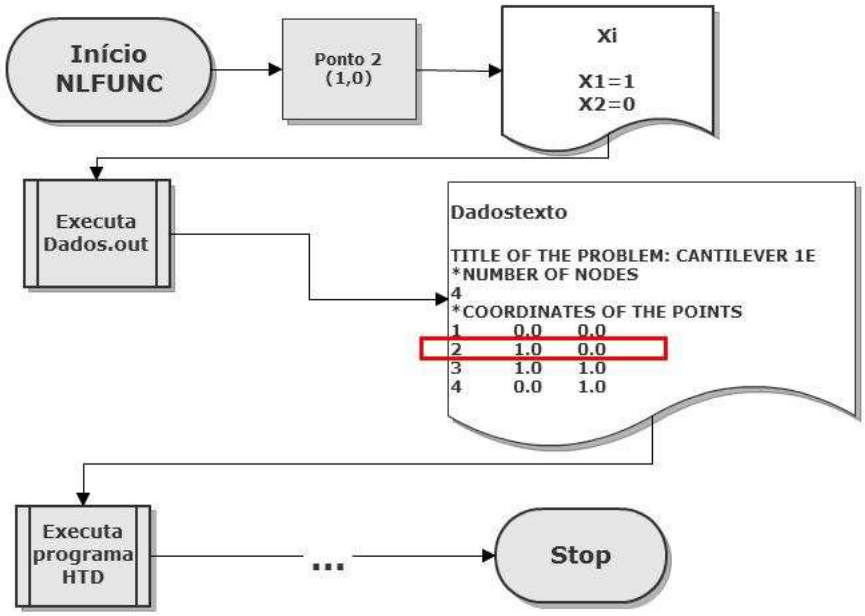


Figura 4.3: Funcionamento da função Dados.out.

do ponto 2 são actualizados os seus valores. No final da execução desta função o ficheiro de entrada necessário para a execução do programa de cálculo HTD está pronto e pode ser utilizado.

4.4 Função Get area

Com a função $get\ area$ pretende-se calcular a área total da estrutura e guardar esse valor num ficheiro ao qual se deu o nome de $areatot$.

Conforme foi referido no final da execução do programa HTD resulta um conjunto de ficheiros. A cada um deles está associado um elemento finito (ficheiros $element\ n$). Juntamente com outras informações é nestes ficheiros que se encontram registados os valores da área para cada um dos elementos. Portanto, o cálculo da área total da estrutura pode ser dada pela soma das áreas de todos os elementos finitos. Na Figura 4.4 é apresentado o algoritmo referente a este processo.

O algoritmo começa por abrir o ficheiro de entrada $Dadostexto$, onde se encontra a informação referente ao número de elementos existentes na estrutura. Posteriormente, é efectuada uma busca em todos os ficheiros referentes aos elementos finitos (elemento n) para obter os valores das suas áreas. A área total obtém-se somando os valores atrás obtidos. Por último, esse valor é guardado no ficheiro $areatot$.

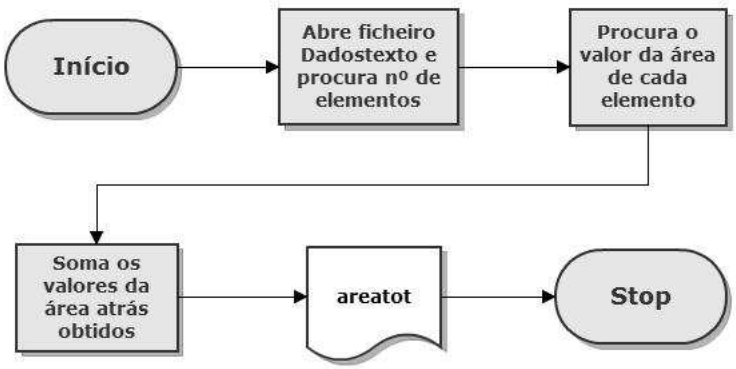


Figura 4.4: Algoritmo Get area.

4.5 Função pos processamento

Esta função apenas tem como objectivo executar de forma automática o pós processamento do programa HTD apresentado na Figura 3.3.

4.6 Função programa HTD

Esta função dá ordem para a execução da função $Dados.out$ que, por sua vez, produz o ficheiro $Dadostexto$ (ficheiro de entrada). No fim executa o programa HTD.

4.7 Função Get tensoes e Get desl

A função $Get\ tensoes$ e $Get\ desl$ têm como objectivo recolher a informação estritamente necessária para definir as funções pertencentes ao problema. A função $Get\ tensoes$ recolhe a informação relativa a tensões, enquanto que a função $Get\ desl$ reúne a informação relativa a deslocamentos.

As subrotinas associadas às duas funções não apresentam diferenças significativas, motivo pelo qual apenas se explicará o algoritmo $Get\ tensoes$.

O algoritmo $Get\ tensoes$, apresentado na Figura 4.5, começa por procurar no ficheiro correspondente à função $pos\ processamento$ o número de pontos para os quais foram determinadas as tensões. De seguida, é aberto o ficheiro que guarda essa informação (denominado por $Stress\ table$), retirada a informação relativa ao estado de tensão e guardada no ficheiro $Tensor$.

Pode-se constatar que a um ponto está associado três valores (σ_{xx}, σ_{yy} e τ_{xy}). Os valores referentes às tensões têm de ser guardados segundo o critério para que depois ao proceder à leitura do ficheiro $Tensor$ não ocorram problemas.

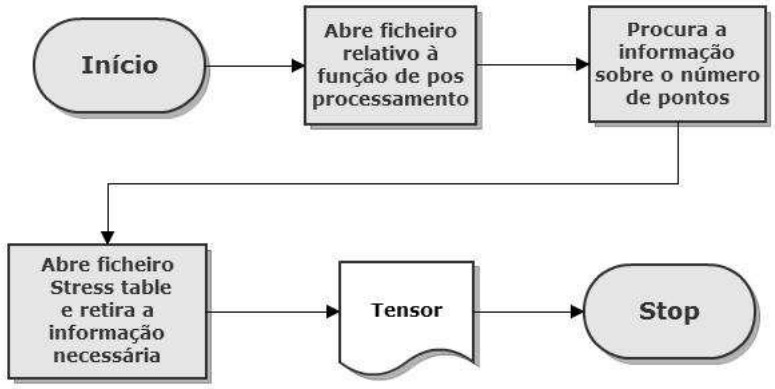


Figura 4.5: Algoritmo Get tensoes.

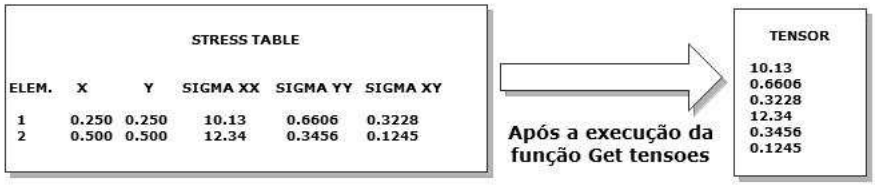


Figura 4.6: Exemplo tipo do ficheiro resultante da execução da função Get tensoes.

Definiu-se, então, que cada linha apenas contem um valor. Assim, as primeiras três linhas correspondem ao primeiro ponto, nas linhas quatro à seis ficam registados os valores do segundo ponto e assim sucessivamente. Na Figura 4.6 apresenta-se um exemplo que ilustra o que se pretende obter após a execução da função $Get\ tensoes$.

4.8 Exemplos numéricos

Foram realizados alguns exemplos práticos a fim de provar que, efectivamente, com a junção do programa de cálculo HTD e o algoritmo NLPQL se obtêm bons resultados. Foi também realizados a optimização das estruturas através do software ANSYS. Para cada exercício foi possível comparar os resultados obtidos através do programa desenvolvido no presente trabalho com os resultados de outros autores e com os resultados do software ANSYS.

4.8.1 Optimização de forma de uma consola

No primeiro exercício pretende-se optimizar a área de uma consola. A consola está sujeita a um carregamento vertical de 0.1 kN/m em toda a fronteira superior. Considerou-se que o material tem um módulo de elasticidade de valor 1kPa e um coeficiente de Poisson de 0.3. Pretende-se neste problema optimizar a forma da fronteira inferior da consola sem que sejam ultrapassados os valores de tensão

estabelecidos e sem alterar a fronteira superior.

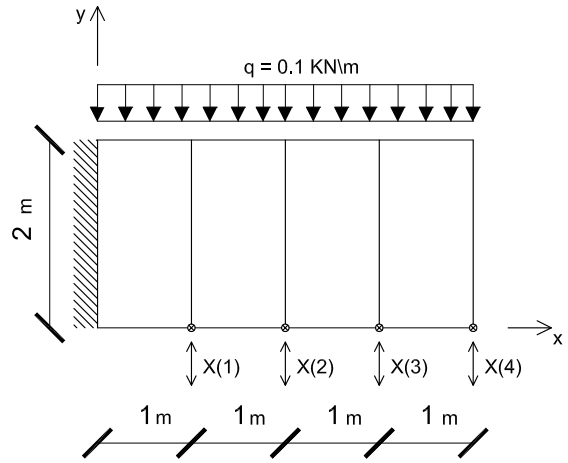


Figura 4.7: Consola sujeita à optimizaçõ da área.

No primeiro teste a consola foi dividida em quatro elementos finitos conforme apresentado na Figura 4.7. Foi utilizado um polinómio de grau 7 ($d_u = 7$) para a aproximação dos deslocamentos e um polinómio de grau 5 ($d_t = 5$) para a aproximação das tracções, totalizando assim 168 graus de liberdade. Ao efectuar esta divisão foi possível definir as coordenadas verticais dos pontos como sendo as variáveis de projecto. Como se pode observar na Figura 4.7 os pontos são os vértices dos elementos finitos. Assim, durante o processo de optimização os pontos podem-se deslocar verticalmente à excepção do ponto mais à esquerda que se encontra estaticamente impedido de se mover. Definiu-se igualmente que estes pontos estariam limitados ao intervalo [-1.0,1.8]. Este intervalo foi escolhido de forma a evitar que as alturas das secções pudessem tomar um valor muito reduzido e para permitir, caso necessário, o aumento da secção da consola.

Como restrição optou-se por limitar a tensão normal segundo a direcção x a um valor de -2.0kPa. Posteriormente, aumentou-se esse valor para -3.0kPa e -4.0kPa. Consideram-se 19 pontos igualmente espaçados na fronteira inferior da consola, sendo que nenhum dele é o ponto com coordenadas (0,0) por se tratar de um ponto de concentração de tensões.

Os resultados obtidos estão presentes na Tabela 4.1 e pode-se verificar graficamente o desenvolvimento das tensões na estrutura na Figura 4.9 quando se restringiu a tensão a -2.0 kPa.

Verifica-se que a optimização ocorreu como esperado. Ao fim de sete iterações, para um valor de σ_{xx} restringido a -2.0 kPa é alcançado o valor óptimo da área cujo valor é de 3.44 m^2. Pode-se igualmente constatar que com o aumento do limite para -3.0 kPa ou -4.0 kPa da tensão normal segundo o eixo x obtém-se um

	$\sigma_{xx} = -2.0kPa$	$\sigma_{xx} = -3.0kPa$	$\sigma_{xx} = -4.0kPa$
nº Iterações	7	6	9
X_1 (m)	0.81	0.99	1.10
X_2 (m)	1.23	1.36	1.45
X_3 (m)	1.62	1.69	1.73
X_4 (m)	1.80	1.80	1.80
A (m^2)	3.44	3.05	2.81
Redução(%)	57.0	61.9	64.9

Tabela 4.1: Optimização da consola - 4 Elementos finitos com polinómios de 1ºgrau na fronteira inferior.

valor de área mais reduzido na consola .

Como foi anteriormente referido, este estudo foi realizado dividindo a consola em quatro elementos. Tendo em conta que as variáveis de projecto são as coordenadas verticais dos pontos da fronteira inferior, o aumento do número de divisões possibilita à estrutura organizar-se melhor e obter áreas mais reduzidas. Assim sendo, procedeu-se à optimização da consola considerando um maior número de elementos. Na Figura 4.8 encontra-se um gráfico representativo da variação da área (A/A_0) quando são realizadas mais divisões (Div) na estrutura.

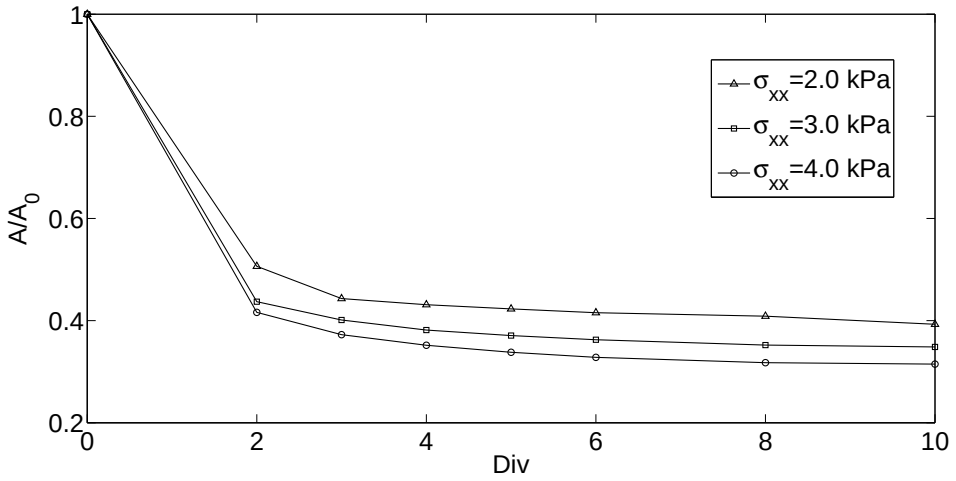
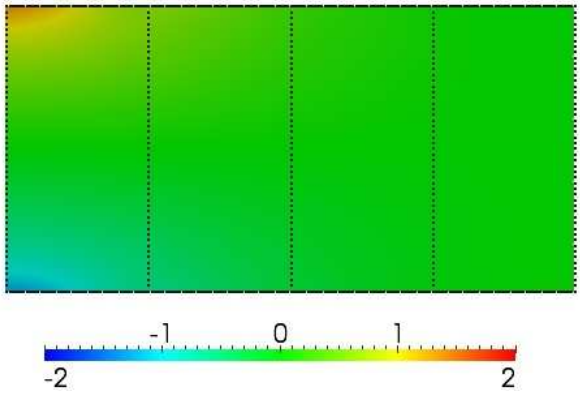
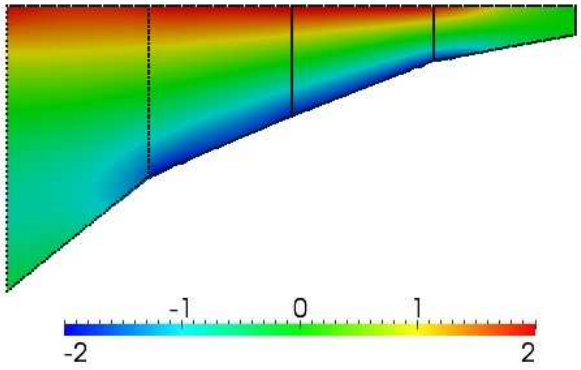


Figura 4.8: Variação da área da consola para varios números de elementos.

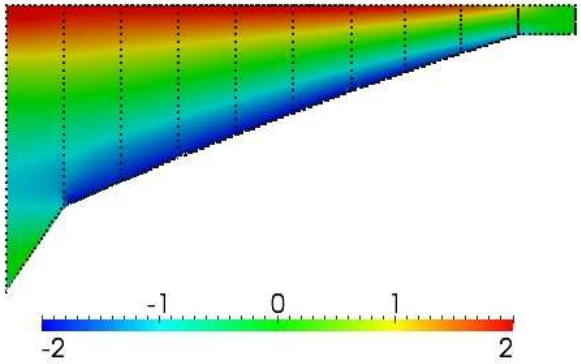
Constata-se que, para qualquer um dos casos, à medida que se aumenta o número de elementos, a área da consola vai diminuindo. Ao aumentar o número de elementos, aumenta-se também o número variáveis e a estrutura ganha mais liberdade para se reorganizar e reduzir a sua área.



(a) consola inicialmente - 4 elementos finitios $(\sigma_{xx}[-2.0; 2.0]\ kPa)$



(b) consola optimizada - 4 elementos finitios $(\sigma_{xx}[-2.0; 2.0]\ kPa)$



(c) consola optimizada - 10 elementos finitios $(\sigma_{xx}[-2.0; 2.0]\ kPa)$

Figura 4.9: Consola antes e após o processo de optimização.

Como se pode verificar aquando do estudo do desempenho dos elementos finitos híbridos-Trefftz, este método possibilita obter bons resultados utilizando na sua formulação poucos elementos. Embora não ponha em causa o desempenho

deste método, a divisão da estrutura num número exagerado de elementos não tira proveito das suas potencialidades.

De forma a tirar proveito das potencialidades do método dos elementos finitos híbridos-Trefftz considerou-se a hipótese da fronteira inferior da consola poder ser composta por um polinómio de terceiro grau. Este tipo de optimização, com recurso a fronteira não lineares, tem sido fruto de investigação por parte de alguns autores como U. Schramm ou V. Braitbant. Os seus trabalhos mostram que o uso de fronteiras definidas por funções polinomiais proporcionam melhores resultados para as estruturas sujeitas à optimização [25, 6].

A utilização deste tipo fronteiras obriga à alteração das variáveis de projecto. Se até agora as variáveis utilizadas correspondiam à coordenada vertical dos vértices dos elementos finitos, de agora em diante será adicionado um novo tipo de variável. Este tipo de variável corresponde aos ângulos tangentes dos pontos extremistas dos elementos finitos.

Assim, procedeu-se novamente à optimização da consola, mas desta vez, recorrendo a um elemento e com a possibilidade da fronteira inferior da consola ser definida por um polinómio de terceiro grau. As variáveis de projecto são apresentadas na Figura 4.10.

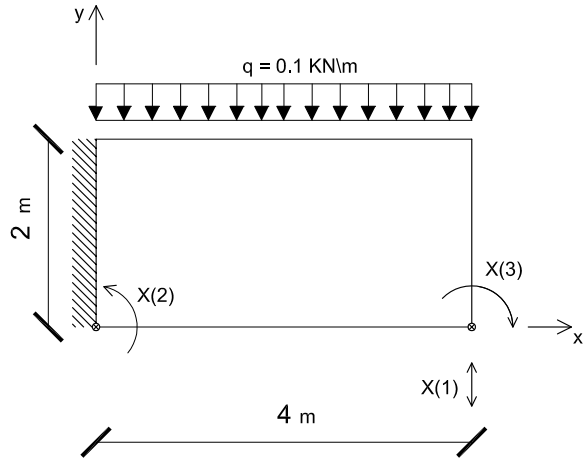


Figura 4.10: Optimização da consola com recurso a um polinómio de 3º grau.

A primeira variável mantem-se restringida ao intervalo $[-1.0, 1.8]$ e as restantes passam a estar limitadas ao intervalo $[-2\pi, 2\pi]$. Todas as restantes definições não foram alteradas. Os resultados do processo de optimização são apresentados na Tabela 4.2.

Comparando os resultados obtidos com os anteriores, aquando da divisão da consola em quatro elementos, verifica-se que praticamente não existe diferença de valores. No primeiro caso, em que se impunha que a tensão normal segundo

	$\sigma_{xx} = -2.0kPa$	$\sigma_{xx} = -3.0kPa$	$\sigma_{xx} = -4.0kPa$
nº Iterações	4	9	5
X_1 (m)	1.8	1.66	1.74
X_2 (rad)	1.52	2.06	1.94
X_3 (rad)	0.12	-0.634	-0.72
A (m^2)	3.42	3.10	2.89
Redução(%)	57.2	61.2	63.8

Tabela 4.2: Optimização da consola - 1 elemento finito com polinómio de 3º grau na fronteira.

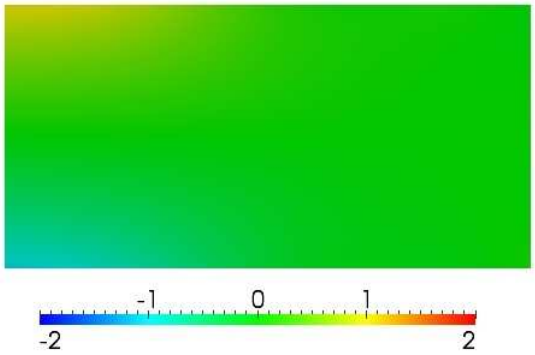
a direcção x não ultrapassasse -2.0 kPa, obteve-se um valor para a área de $3.42m^2$, menos 0.5% do que o valor de área obtido se for utilizada uma divisão de quatro elementos. Quando se aumentou o limite do critério para valores de -3.0 ou 4.0 kPa os resultados não ultrapassaram erros de 3%. Como se pode entender, a ordem de grandeza destes erros tornam-se insignificantes. Pode-se, assim, concluir que utilizando um elemento finito com a fronteira definida por um polinómio terceiro grau se consegue obter os mesmos resultados considerando uma divisão da consola em quatro elementos. Na Figura 4.11 é apresentada a consola optimizada recorrendo a um polinómio de terceiro grau.

As optimizações apresentadas anteriormente apenas serviram para, por um lado verificar o desempenho do programa desenvolvido e, por outro lado justificar a utilização de polinómios na definição das fronteiras dos elementos finitos. Na verdade, na realização do dimensionamento de uma peça estrutural não é apenas a tensão normal segundo a direcção x que se apresenta como relevante. Uma estrutura poderá apresentar um bom desempenho para a tensão segundo uma direcção, mas poderá ocorrer grandes problemas segundo outra. Num processo de dimensionamento de uma estrutura são utilizados outros meios para verificar a segurança da estrutura. Normalmente é utilizada a tensão de Von Misses que relaciona todas tensões a que a estrutura esta sujeita. Depois com a obtenção da tensão de Von Mises poderá verificar-se se todos os pontos pertencentes à estrutura respeitam o critério imposto, que normalmente corresponde à própria tensão de cedência do material. Neste trabalho procedeu-se à optimização da consola recorrendo à tensão de Von Mises que, para o caso do estado de tensão plana, é expressa através da equação 4.2.

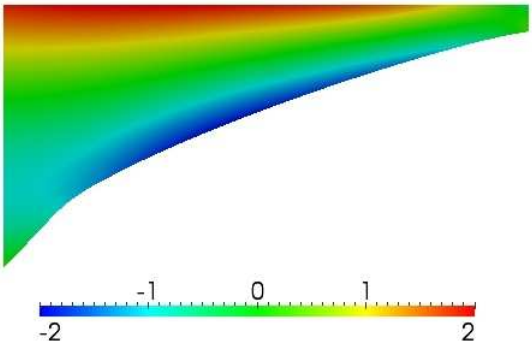
$$\sigma^{VM} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\tau_{xy}^2} \tag{4.2}$$

Esta estrutura já foi analisada por Teixeira e Cismasiu [16], onde se procedeu à optimização de forma de uma consola composta na sua fronteira por um polinómio de terceiro grau.

Os parâmetros referentes à geometria da peça, o carregamento e o módulo de elasticidade foram os mesmos que os utilizados nas optimizações realizadas anteriormente, à excepção do valor do coeficiente de Poisson. Tinha-se



(a) consola inicialmente $(\sigma_{xx}[-2.0; 2.0]\ kPa)$



(b) consola optimizada $(\sigma_{xx}[-2.0; 2.0]\ kPa)$

Figura 4.11: Consola antes e após a optimização utilizando um polinómio de 3º grau

anteriormente arbitrado um valor que $\upsilon = 0.3$, mas como se pretende comparar os resultados obtidos com os mencionados no artigo [16], alterou-se o valor deste coeficiente para $\upsilon = 0$ (valor utilizado no artigo).

Na optimização foi considerado um elemento finito híbrido-Trefftz. A aproximação das tracções fez-se através de um polinómio de sétimo grau ($d_t = 7$) e para a aproximação dos deslocamentos utilizou-se um polinómio de nono grau ($d_u = 9$), perfazendo assim 54 graus de liberdade.

O controlo dos valores da tensão de Von Misses na fronteira inferior da consola foi feita através de 21 pontos, conforme apresentado na Figura 4.12, e não podem ultrapassar o valor de 1.2 kPa (valor utilizado em [16]). Foram necessárias 14 iterações para concluir a optimização da consola. Os resultados são apresentados na Tabela 4.3.

Posteriormente, realizou-se a optimização da consola no software ANSYS. Para tal, a estrutura foi modelada através de elementos quadriculares de oito nós (Plane82) e foram definidas como variáveis de projecto os pontos pertencentes

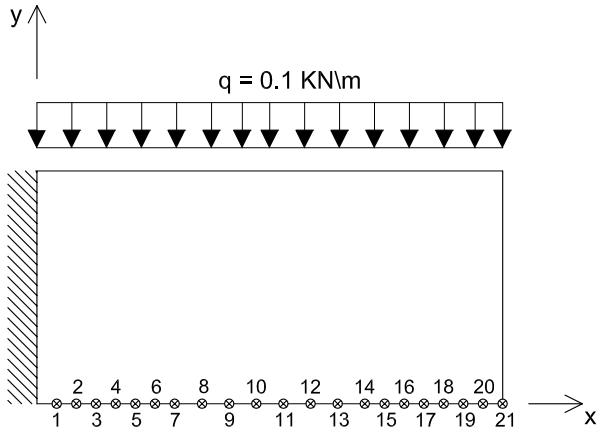


Figura 4.12: Consola sujeita a optimização de forma - Pontos que efectuam o controlo da tensão na fronteira.

	$\sigma^{VM} = 1.2kPa$
nº Iterações	14
X_1 (m)	1.80
X_2 (rad)	0.14
X_3 (rad)	0.15
A (m^2)	4.4
Redução(%)	45.0

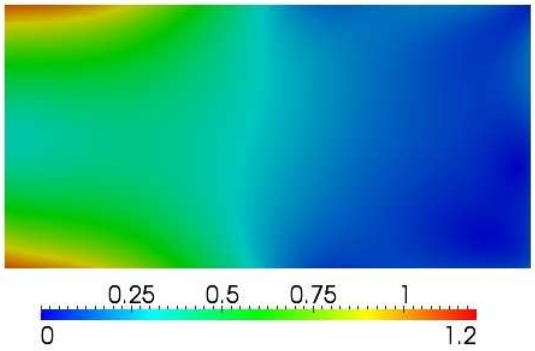
Tabela 4.3: Optimização de forma da consola Resultados obtidos utilizando a tensão de Von Mises.

à fronteira inferior da consola. As características dos materiais e a geometria da estrutura não foram alteradas. A redução de área obtida através do software ANSYS, do programa desenvolvido neste trabalho, e os mencionados no artigo [16] são apresentados na Tabela 4.4. A representação gráfica da consola pode ser consultada na Figura 4.13.

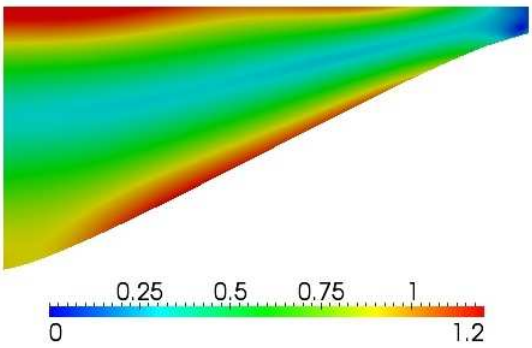
	HTD	Freitas [16]	ANSYS
Nº Iterações	11	9	13
Nº Elementos	1	2	76
d_u	9	7	-
d_t	7	5	-
N	54	84	532
Área (m^2)	4.40	-	4.41
Redução(%)	45.0	47.4	44.9

Tabela 4.4: Optimização de forma da consola - Comparação de resultados .

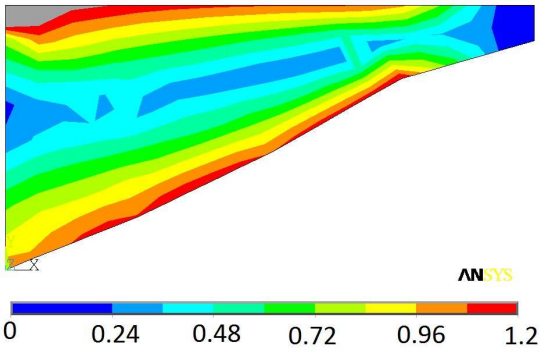
Como se pode verificar, em qualquer um dos casos em que se tenha utilizado elementos finitos híbridos-Trefftz conseguiu-se com um número mais reduzido de elementos e de graus de liberdade atingir os mesmos resultados que o software



(a) consola inicialmente $\left(\sigma^{VM}[0;1.2]\ kPa\right)$



(b) Consola optimizada através do HTD $\left(\sigma^{VM}[0;1.2]\ kPa\right)$



(c) Consola optimizada através do ANSYS $\left(\sigma^{VM}[0;1.2]\ kPa\right)$

Figura 4.13: Consola antes e após a sua optimização limitando o valor da tensão de Von Mises.

ANSYS. Os resultados obtidos por Teixeira e Cismasiu [16] são, efectivamente, melhores face aos obtidos neste trabalho. Contudo, utilizam mais elementos finitos híbridos-Trefftz, um número mais elevado de graus de liberdade assim como, as expressões analíticas das funções associadas ao problemas. No final apenas reduzem em mais 2.4 % a área. Assim, pode-se concluir que os resultados obtidos neste traballho são bastantes satisfatórios.

4.8.2 Optimização de uma placa composta por uma aresta inclinada

Este exercício foi retirado do artigo desenvolvido por Schramm e Pilkey [24], que visa a optimização de forma utilizando fronteiras não lineares e recorrendo à formulação convencional dos elementos finitos. Posteriormente, este artigo também foi utilizado na investigação desenvolvida por Teixeira e Cismasiu [16] onde se pretendia a utilização de elementos finitos híbridos-Trefftz no processo de optimização.

Trata-se de um placa sujeita a uma força de tracção conforme apresentada na Figura 4.15. O material constituinte da estrutura tem um módulo de elasticidade igual 2.26x10^5 MPa e um coeficiente de Poisson de 0.3.

.

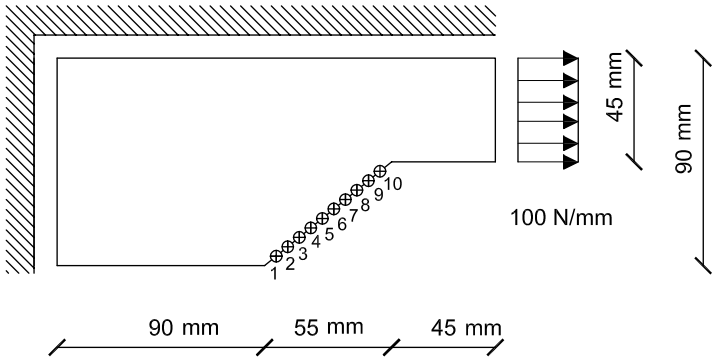


Figura 4.14: Placa sujeita a uma força de tracção.

A placa é composta por uma aresta inclinada sobre a qual se procederá à optimização. Pretende-se que, sem alterar as posições dos vértices da estrutura, reduzir a área (cujo o valor inicial é de 14512.5 mm^2) sem que seja ultrapassado um valor da tensão de Von Mises de 120 MPa. Para tal, é utilizado um polinómio de terceiro grau para definir a fronteira da placa. As variáveis mais uma vez são os ângulos tangentes aos vértices da aresta e estão limitadas ao intervalo $[0, \pi]$. O controlo do valor da tensão ao longo da aresta é feito através de 10 pontos, que se encontram identificados na figura.

Para modelar a estrutura foi considerado apenas um elemento finito, e foi utilizado um polinómio de grau 9 para aproximar os deslocamentos ($d_u = 9$) e um polinómio de grau 5 ($d_t = 5$) para aproximar as tracções, originando assim 50 graus de liberdade.

Os resultados obtidos neste trabalho, assim como os mencionados nos artigos [24, 16] e ainda os obtidos através no software ANSYS, são apresentados na Tabela 4.5.

Como se pode verificar, os resultados obtidos neste trabalho são semelhantes

	HTD	Freitas [16]	Schramm[24]		ANSYS
Nº Iterações	11	10	9	18	23
Nº Elementos	1	3	300	300	102
d_u	9	9	-	-	-
d_t	5	5	-	-	-
N	50	162	2322	2322	718
Nº variáveis	2	2	4	8	3
Área inicial (mm^2)	14512.5				
Área reduzida (mm^2)	13469.1	-	13376.4	13355.9	13667
Redução (%)	7.2	7.3	7.8	8.0	5.8

Tabela 4.5: Optimização da Placa- Comparacão de resultados.

aos resultados obtidos através de outros autores. Contudo, em qualquer um dos estudos mencionados foram utilizados mais elementos finitos e um número de graus de liberdade mais elevados. Os melhores resultados foram obtidos por Schramm e Pilkey [24] que ao longo trabalho procederam ao aumento do número de variáveis tentando obter melhorias nos resultados que, como se pode verificar, não se distanciam muito dos restantes casos. Assim, pode-se concluir que os resultados obtidos neste trabalho são bastante satisfatórios. Nas Figura 4.15 e Figura 4.16 pode-se observar a representação gráfica dos resultados obtidos.

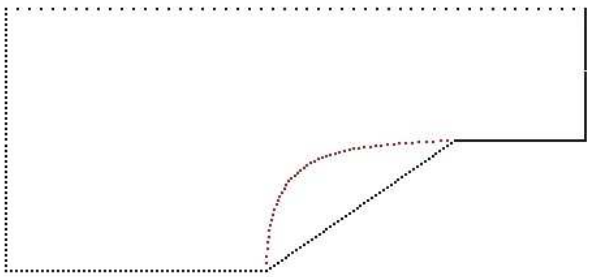
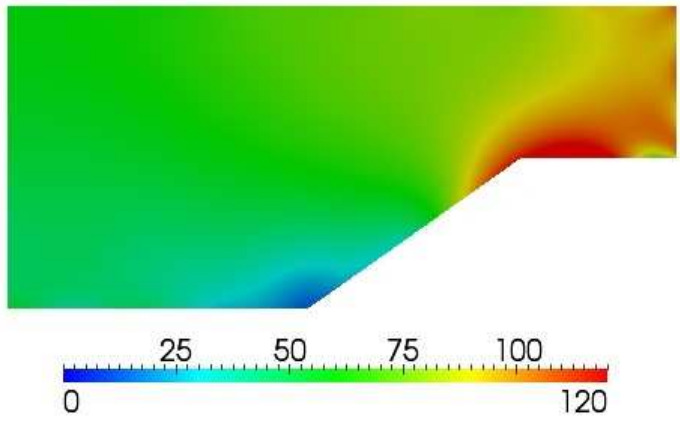
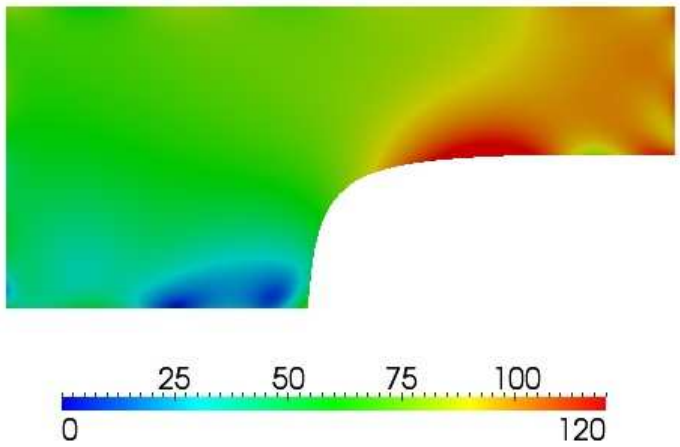


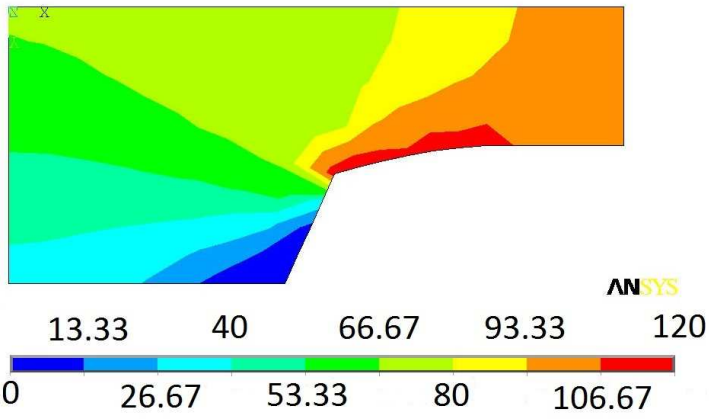
Figura 4.15: Representação da placa antes e após a optimização.



(a) Estrutura inicialmente $\left(\sigma^{VM}[0;120]\ MPa\right)$



(b) Estrutura optimizada $\left(\sigma^{VM}[0;120]\ MPa\right)$



(c) Estrutura optimizada através do ANSYS $\left(\sigma^{VM}[0;120]\ MPa\right)$.

Figura 4.16: Optimização da placa - Representação gráfica dos resultados obtidos.

4.8.3 Optimização de forma de uma viga bi-apoiada

O último problema teste apresentado neste trabalho prende-se com a optimização de uma viga simplesmente apoiada. Este exercício ao contrário dos anteriores não foi retirado de nenhuma referência bibliográfica, no entanto, optou-se por o realizar. Os resultados finais serão comparados com os obtidos através do software ANSYS.

A viga em estudo é composta por aço S235, o que corresponde a um módulo de elasticidade de 210 GPa, um coeficiente de Poisson de 0.3 e uma tensão de cedência de 235 Mpa. Trata-se de um viga com 3 m de comprimento e 1 m de largura, conforme apresentado na Figura 4.17. Pretende-se com este problema minimizar a área da viga sem que a sua fronteira inferior não ultrapasse o valor de cedência do material. Na análise da viga recorreu-se mais uma vez à tensão de Von Misses. Assim, ao longo da fronteira foram considerados 11 pontos onde se procedeu ao controlo das tensões. É importante referir que nenhum destes pontos são os pontos extremos do elemento, pois são pontos de concentração de tensão. Por se tratar de uma estrutura simétrica apenas foi considerada uma variável que corresponde ao ângulo tangente do ponto correspondente ao apoio. Os pontos fronteiriços onde são impostas as restrições apenas são contabilizados até meio da viga.

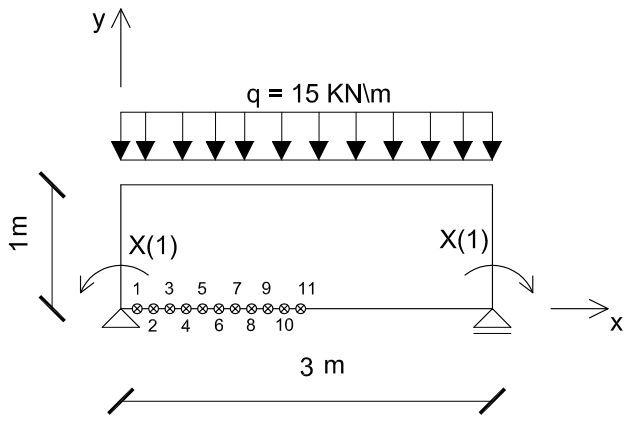


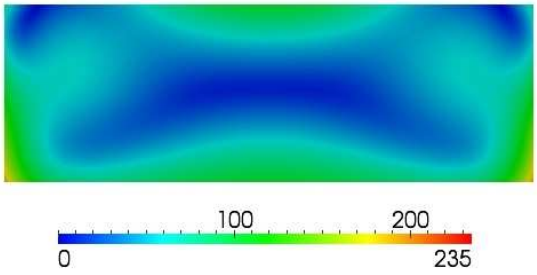
Figura 4.17: Viga sujeita à optimização de forma.

Mais uma vez foi utilizado apenas um elemento finito. A aproximação dos deslocamentos fez-se através de um polinómio de grau sete ($d_u = 7$), enquanto que a aproximação das tracções foi feita através de um polinómio de primeiro grau dando origem a 33 graus de liberdade. Ao fim de 9 iterações o processo de optimização terminou. A mesma estrutura foi sujeita ao mesmo tipo de optimização através do software ANSYS. Na modelação da viga foram considerados elementos quadriculares de oito nós (PLANE82) e as variáveis de projecto consideradas foram os pontos pertencentes à fronteira inferior da viga. Os resultados são apresentados na Tabela 4.6 e na Figura 4.18.

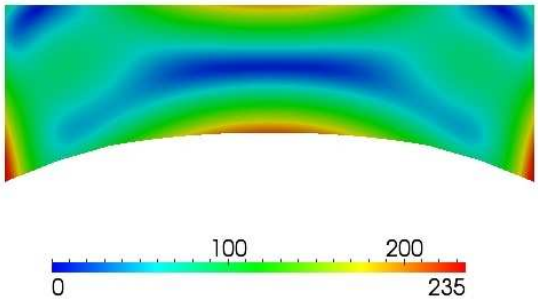
	HTD	ANSYS
Nº Iterações	9	11
Nº Elementos	1	52
N	33	390
A (m^2)	2.20	2.41
Redução(%)	26.7	19.7

Tabela 4.6: Resultados obtidos para a tensão de Von Mises.

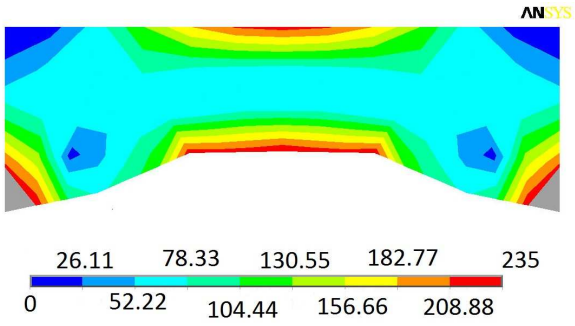
Como se pode verificar através do ANSYS foram obtidos piores resultados. A utilização de um polinómio de segundo grau na definição da fronteira na viga proporcionou que a estrutura tivesse mais capacidade para se poder reorganizar, sendo possível obter uma melhor definição da fronteira. Constata-se que a estrutura obtida através do software ANSYS tende a ter a mesma configuração na parte inferior da viga. No entanto, não atinge a mesma perfeição que a estrutura optimizada através dos elementos finito híbridos-Trefftz.



(a) Viga inicialmente $\left(\sigma^{VM}[0;235]\ MPa\right)$



(b) Viga optimizada através do HTD $\left(\sigma^{VM}[0;235]\ MPa\right)$



(c) Viga optimizada através do ANSYS $\left(\sigma^{VM}[0;235]\ MPa\right)$

Figura 4.18: Viga optimizada através do software ANSYS e do HTD.

Capítulo 5

Conclusões e desenvolvimentos futuros

No presente trabalho criou-se um programa que procedesse à optimização de estruturas, recorrendo a elementos finitos híbridos-Trefftz. A criação do programa baseou-se em articular um algoritmo que procede à resolução de problemas de optimização com um outro programa que resolve problemas através da formulação híbrida Trefftz. Pretendia-se assim que, com um número reduzido de informações dadas pelo utilizador, fosse possível obter estruturas optimizadas tirando partido das potencialidades que esta formulação oferece. Depois da realização do programa procedeu-se à optimização de algumas estruturas teste com o objectivo de verificar se o programa desenvolvido correspondia ao esperado. Por fim, os resultados obtidos foram comparados com os resultados de outros autores e, ainda, com os resultados no software ANSYS.

Apresentam-se, de seguida, as principais conclusões deste trabalho e sugerem-se alguns possíveis desenvolvimentos futuros.

5.1 Principais conclusões

Com a realização deste trabalho concluir-se que ao integrar o programa HTD com o NLPQL são obtidos bons resultados. A utilização de elementos finitos híbridos-Trefftz proporcionou uma redução significativa de graus de liberdade e a diminuição dos elementos finitos necessários para modelar as estruturas. A hipótese de se poder definir as fronteiras dos elementos através de polinómios de grau superior acrescem mais-valias para a optimização de estruturas. Como se pode verificar, ao longo do trabalho, foi utilizado apenas 1 elemento finito que garantiu resultados bastantes satisfatórios nas optimizações realizadas. Constata-se, igualmente, que o uso de polinómios não lineares possibilitaram definir melhor a configuração das fronteiras o que não se verificou nos resultados obtidos através do software ANSYS. Possivelmente os resultados obtidos através do ANSYS poderiam ser melhorados, mas seria necessário considerar mais elementos finitos e consequentemente aumentar o número de graus de liberdade.

Outra vantagem no uso da formulação híbrida-Trefftz é a menor preocupação na malha. A cada iteração do processo de optimização, o software ANSYS elimina alguns elementos finitos pertencentes à estrutura e renumera os restantes. Assim, como é obrigado a renumerar os pontos pertencentes à estrutura. Este processo obriga a cada iteração o cuidado de reformular a malha e atribuir novos números aos elementos e aos pontos. Acresce ainda o trabalho de não "esquecer"quais os pontos sobre os quais são feitas as restrições. Este aspecto é facilmente esquecido pelo utilizador, mas gera cuidados adicionais aos programadores. Esses cuidados não são necessários ter nos elementos finitos híbridos-Trefftz, uma vez que as estruturas já utilizam um número muito reduzido de elementos.

Por fim, e apenas a título de curiosidade, verificou-se que o número de iterações utilizadas pelo programa desenvolvido neste trabalho nunca foi superior ao software ANSYS, o que mostra o bom desempenho do algoritmo NLPQL.

Assim sendo, pode-se concluir que o uso de elementos finitos híbridos conjugados com o algoritmo NLPQL apresentam resultados muito positivos. Através de um número reduzido de elementos e de graus de liberdade conseguem-se alcançar bons resultados.

5.2 Desenvolvimentos Futuros

Para possíveis estudos futuros propõe-se:

- Proceder à optimização de estruturas com outras características como, por exemplo, estruturas compostas com orifícios ou com geometrias irregulares;
- Optimizar estruturas recorrendo a elementos finitos híbridos-Trefftz de tensão e comparar os resultados obtidos;
- Optimizar de estruturas de aço ou de betão armado e realizar ensaios em laboratório a fim de analisar se efectivamente apresentam o desempenho previsto.

Referências Bibliográficas

[1] *University of Alberta (Canadá): ANSYS Tutorials - 2001*, Consultado a Junho de 2012. `http://www.mece.ualberta.ca/tutorials/ansys/`.

[2] *Paraview*, Consultados a Julho de 2012. `http://www.paraview.org/`.

[3] *Grupo de investigação de Análise Estrutural Universidade Técnica de Lisboa*, Consultados a Maio de 2012. `http://www.civil.ist.utl.pt/HybridTrefftz/`.

[4] Babu, R.: *Numerical Methods*. Pearson Education, 2010.

[5] Bendsoe, W. P. e O. Sigmund: *Topology Optimization: Theory, Methods, and Applications*. Springer, 2003.

[6] Braitbant, V. e C. Fleury: *Shape Optimization Design Using B-Splines*. Computer Metods In Applied Mechanics and Engineering, 44:247–267, 1984.

[7] Cardoso, J. M. B., P. S. G. Coelho e J. R. ALmeida: *Actas do V Congreso de Métodos Numéricos en Ingenieria, eds. J.M. Goicolea, C. Mota Soares, M.Pastor, G. Bugeda, SEMNI, Madrid, Espanha.* 2002.

[8] Chaudhuri, A. B.: *The Art of Programming Through Flowcharts & Algorithms*. Laxmi Publications Pvt Ltd, 2005.

[9] Christensen, P. W. e A. Klarbring: *An Introduction to Structural Optimization*. Springer, 2008.

[10] Cismasiu, C.: *The Hybrid-Trefftz Displacement Element For Static and Dynamic Structural Analysis Problems*. Tese de Doutoramento, Universidade Técnica de Lisboa - Instituto Superior Técnico, Setembro 2000.

[11] Computers and Structures, Inc.: *SAP2000 features a to z problems*, 2007.

[12] Correia, F. N., L. Tavares R. Oliveira e I. Hall Themido: *Investigação Operacional*. Mc Graw-Hill, 1996.

[13] Farkas, J.: *Analysis and Optimum Design of Metal Structures*. Taylor & Francis, 1997.

[14] Forsgren, A., P. Gill e M. Wright: *Interior Methods for Nonlinear Optimization*, volume 44. Society for Industrial and Applied Mathematics, 2002.

[15] Freitas, J. A. T. e C. Cismasiu: *Developments with hybrid-Trefftz stress and displacement elements*. Journal of Structural Engineering, 8:289 – 311, 2001.

[16] Freitas, J. A. T. e I. Cismasiu: *Shape optimization with hybrid-Trefftz displacement elements.* International Journal For Numerical Methods In Engineering, 53:473–498, 2001.

[17] Haftka, R. T. e Z. Gürdal: *Elements of Structural Optimization*. Kluwer Academic Publishers, 1992.

[18] Hörnlein, H.R.E.M. e K. Schittkowski: *Software Systems for Structural Optimization*. International Series of Numerical Mathematics. Birkhäuser, 1993.

[19] Jain, M. K. e S. R. K. IYENGAR: *Numerical Methods For Scientific And Engineering Computation*. New Age International Publishers, 2007.

[20] Jirousek, J.: *Basis for development of large finite elements locally satisfying all field equations*. Computational Methods in Applied Mechanical Engineering, 14:65–92, 1978.

[21] Mathworks: *Tutorial for Optimization Tool Examples*, Consultado em Julho de 2012. `www.mathworks.com/help/toolbox/optim/ug/bq06zgs.html`.

[22] Schittkowski, K.: *NLPQLP: A Fortran Implementation of a Sequential Quadratic Programming Algorithm for Parallel Computing*. Departament of Computer Science University of Bayreuth.

[23] Schittkowski, K.: *NLPQL: A Fortran subroutine solving constrained nonlinear programming problems*. Annuals of operations research 5, V:485–500, 1985/6.

[24] Schramm, U. e W. D. Pilkey: *The Application Of Rational B-Splines For Shape Optimization*. The First World Congress Of Structural And Multidisplinary Optimization., 1995.

[25] Schramm, U. e W. Pilkey: *Higher Order Boundary Elements For Shape Optimization Using Rational B-Splines*. Engineering Analysis with Boundary Elements, 14:255–266, 1994.

[26] Topping, B. H. V.: *Shape Optimization of Skeletal Structures: A Review*. Journal of Structural Engineering, 109:1933–1951, 1983.

Apêndice A

Exercícios teste

A.1 Método de Lagrange e condições de Karush-Kuhn-Tucker (KKT)

Para avaliar o desempenho do algoritmo NLPQL realizaram-se dois problemas que possibilitaram sem grande complexidade determinar a sua solução analítica. O primeiro associado à maximização da rigidez de uma estrutura, encontra-se no subcapítulo 2.4.2. O segundo exercício, referente à minimização do peso de uma viga consola, apenas se encontra resolvido para o caso em que a estrutura apenas se encontra dividida em 2 elementos. De forma a que se consiga ter mais casos de estudo, aumentou-se o número de divisões efectuadas na cansola. Para a sua resolução analítica foi utilizado o método dos multiplicadores de Lagrange e as condições de Karush-Kuhn-Tucker (KKT). Este método aqui apresentado foi consultado em [9]. Admitindo um problema de optimização composto apenas por restrições de igualdade, tem-se:

$$\begin{cases} Min \quad f(X) \\ s.a \quad \sum_{i=1}^{N} g_i(X) = b_i \end{cases}$$

onde $f(X)$ e $g_i(X)$ representam a função objectivo e as restrições, respectivamente. X é o vector das variáveis integrantes do problema de tamanho M e com N número de restrições de igualdade do problema.

O método de Lagrange baseia-se em estudar uma função designada por função de Lagrange, e com isso determinar a solução analítica do problema. A função de Lagrange é definida:

$$\mathcal{L}(X, \lambda) = f(X) - \sum_{i=1}^{N} \lambda_i (g_i(X) - b_i) \tag{A.1}$$

Como se pode verificar na função de Lagrange são introduzidas novas variáveis (λ_i), designadas por multiplicadores de Lagrange. O problema passa a ser tratado como se não houvessem restrições considerando o gradiente nulo. Entende-se por gradiente como a diferenciação da equação de Lagrange $\mathcal{L}(X, \lambda)$ em relação

a X e λ.

$$\frac{\partial \mathcal{L}}{\partial X_j} = \frac{\partial f(X)}{\partial X_j} - \sum_i^N \lambda_i \frac{\partial g_i(X)}{\partial X_j} = 0 \tag{A.2}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_i} = -(g_i(X) - b_i) = 0 \tag{A.3}$$

As condições anteriormente referidas permitem obter um conjunto de expressões (N+M recordando que N é o numero de restrições e M o número de variáveis), que têm de ser satisfeitas simultaneamente. Desta forma, relacionando as expressões anteriores, consegue-se determinar os valores optimizados do problema.

Contudo, tem de se averiguar se os resultados obtidos na optimização se tratam efectivamente do mínimo global. Atendendo que a segunda derivada indica a forma da função em causa é útil o seu cálculo de forma a indicar se o valor óptimo do problema é local ou global. Assim, se a designada matriz Hessiana for composta por apenas elementos definidos e positivos está-se perante o valor mínimo do problema. Contrariamente, se se estiver a estudar um problema de maximização, o valor óptimo é descoberto somente se todos os elementos pertencentes à matriz Hessiana forem negativos. O cálculo da matriz Hessiana é apresentado de seguida e baseia-se na determinação das derivadas parciais para todas as variáveis existentes na função de Lagrange, conforme é apresentado em (A.4).

$$\begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial x_1^2} & \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_n} \\ \frac{\partial^2 \mathcal{L}}{\partial x_2 \partial x_1} & \frac{\partial^2 \mathcal{L}}{\partial x_2^2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \mathcal{L}}{\partial x_n \partial x_1} & \frac{\partial^2 \mathcal{L}}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 \mathcal{L}}{\partial x_n^2} \end{bmatrix} \tag{A.4}$$

O processo atrás explicito apenas é valido para o caso das restrições de igualdades. Caso se esteja a trabalhar com restrições de desigualdade terá de adicionar algumas condições. Estas condições fazem parte de um trabalho realizado por Harold Kuhn e Albert Tucker e são conhecidas como condições Karush-Kuhn-Tucker ou KKT. São elas:

$$\frac{\partial \mathcal{L}(x,\lambda)}{\partial x_j} \leq 0 \quad se \quad x_j = x_j^{max} \tag{A.5}$$

$$\frac{\partial \mathcal{L}(x,\lambda)}{\partial x_j} = 0 \quad se \quad x_j^{min} < x_j < x_j^{max} \tag{A.6}$$

$$\frac{\partial \mathcal{L}(x,\lambda)}{\partial x_j} \leq 0 \quad se \quad x_j = x_j^{max} \tag{A.7}$$

$$\lambda_i g_i(x) = 0 \tag{A.8}$$

$$g_i(x) \leq 0 \tag{A.9}$$

$$\lambda_i \geq 0 \tag{A.10}$$

$$x \in \chi \tag{A.11}$$

onde $j = 1, ..., M$ e $i = 1, ..., N$.

A.2 Problema de minimização do peso da consola

No subcapítulo 2.4.1 foi resolvido o problema referente à minimização do peso de uma consola quando esta era constituída por dois elementos. Neste texto é apresentada a solução para o caso em que a consola é dividida em 3, 5 e 10 elementos. Na resolução do problema foi utilizado o método de Lagrange recorrendo implementando as condições de Karush-Kuhn-Tucker (KKT). Assim, e relembrando, para uma divisão em três elementos obteve-se o seguinte problema de optimização:

$$\begin{cases} Min \quad f(x_1, x_2, x_3) = C_1(x_1 + x_2 + x_3) \\ \qquad s.a \begin{cases} \frac{1}{x_1^3} + \frac{7}{x_2^3} + \frac{19}{x_3^3} - C_2 \leq 0 \\ x_1 > 0 \\ x_2 > 0 \\ x_3 > 0 \end{cases} \end{cases} \tag{A.12}$$

Com $C_1 = 4\rho Lt$ e $C_2 = \left(2tE \setminus FL^3\right)\delta_0$. Recorrendo ao teorema de Lagrange começou-se por definir a função de Lagrange:

$$\mathcal{L} = C_1(x_1 + x_2 + x_3) + \lambda_1 \left(\frac{1}{x_1^3} + \frac{7}{x_2^3} + \frac{19}{x_1^3} - C_2\right) \tag{A.13}$$

O uso do método de Lagrange só é permitido se se verificar as condições atrás referidas. Analisando o problema,terá de se impor que todos os multiplicadores de lagrange serão positivos, cumprindo o estabelecido em (A.10), as restrições impostas são desigualdades do tipo $g_i(x) \leq 0$, vindo ao encontro do estabelecido em (A.9). Assim, recorrendo mais uma vez à equação de lagrange, (A.13) estabelece-se que:

$$\frac{\partial \mathcal{L}}{\partial x_1} = C_1 + \lambda_1 \left(-\frac{3}{x_1^4}\right) = 0 \tag{A.14}$$

$$\frac{\partial \mathcal{L}}{\partial x_2} = C_1 + \lambda_1 \left(-\frac{21}{x_2^4}\right) = 0 \tag{A.15}$$

$$\frac{\partial \mathcal{L}}{\partial x_3} = C_1 + \lambda_1 \left(-\frac{57}{x_3^4} \right) = 0 \tag{A.16}$$

Organizando os resultados anteriormente obtidos chega-se ao seguinte sistema de equações:

$$C_1 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \lambda_1 \begin{bmatrix} -\frac{3}{x_1^4} \\ -\frac{21}{x_2^4} \\ -\frac{57}{x_3^4} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{A.17}$$

Resolvendo-o confirma-se que o multiplicador de Lagrange toma valor positivo.

$$\begin{cases} \lambda_1 = \frac{C_1 x_1^4}{3} \\ \lambda_1 = \frac{C_1 x_2^4}{21} \\ \lambda_1 = \frac{C_1 x_3^4}{57} \end{cases} \tag{A.18}$$

Relacionando os valores obtidos para o multiplicador de lagrange, obteve-se as seguintes relacões entre as variáveis em causa:

$$\begin{cases} x_2 = 7^{\frac{1}{4}} x_1 \\ x_3 = 19^{\frac{1}{4}} x_1 \end{cases} \tag{A.19}$$

Retomando a condição (A.8) referente aos critérios KKT, é possível aplicar as relações anteriormente determinadas e assim obter a solução óptima para a variável x_1.

$$\lambda_1 \left(g_1(A) \right) = 0 \Leftrightarrow \frac{1}{x_1^3} + \frac{7}{x_2^3} + \frac{19}{x_3^3} - C_2 = 0 \tag{A.20}$$

$$\Leftrightarrow \frac{1}{x_1^3} + \frac{7}{7^{\frac{3}{4}} x_1^3} + \frac{19}{19^{\frac{3}{4}} x_1^3} = C_2 \tag{A.21}$$

$$x_1^* = \left(\frac{1 + 7^{\frac{1}{4}} + 19^{\frac{1}{4}}}{C_2} \right)^{\frac{1}{3}} \tag{A.22}$$

Determinada a solução óptima de uma variável, e recuperando as igualdades obtidas em (A.19) chega-se ao resultado optimizado das restantes variáveis:

$$x_2^* = 7^{\frac{1}{4}} x_1 = 7^{\frac{1}{4}} \left(\frac{1 + 7^{\frac{1}{4}} + 19^{\frac{1}{4}}}{C_2} \right)^{\frac{1}{3}} \tag{A.23}$$

$$x_3^* = 19^{\frac{1}{4}} x_1 = 19^{\frac{1}{4}} \left(\frac{1 + 7^{\frac{1}{4}} + 19^{\frac{1}{4}}}{C_2} \right)^{\frac{1}{3}} \tag{A.24}$$

Agora com os resultados das variáveis optimizado obtém-se a solução óptima do problema:

$$f(x_1, x_2, x_3) = C_1 \left(1 + 7^{\frac{1}{4}} + 19^{\frac{1}{4}}\right) \left(\frac{1 + 7^{\frac{1}{4}} + 19^{\frac{1}{4}}}{C_2}\right)^{\frac{1}{3}} \tag{A.25}$$

Depois da obtenção do valor óptimo do problema é necessário determinar a matriz Hessiana de forma a descobrir se o valor obtido se trata efectivamente do mínimo global do problema. Para o problema em estudo foram obtidos os seguintes resultados:

$$\begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial x_1^2} & \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_2} & \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_3} \\ \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_2} & \frac{\partial^2 \mathcal{L}}{\partial x_2^2} & \frac{\partial^2 \mathcal{L}}{\partial x_2 \partial x_3} \\ \frac{\partial^2 \mathcal{L}}{\partial x_1 \partial x_3} & \frac{\partial^2 \mathcal{L}}{\partial x_2 \partial x_3} & \frac{\partial^2 \mathcal{L}}{\partial x_3^2} \end{bmatrix} = \begin{bmatrix} \frac{\lambda_1 12}{x_1^5} & 0 & 0 \\ 0 & \frac{\lambda_1 84}{x_2^5} & 0 \\ 0 & 0 & \frac{\lambda_1 228}{x_3^5} \end{bmatrix} \tag{A.26}$$

Todos os valores pertencentes à matriz Hessiana são nulos ou positivos, o que indica que a variação da taxa de crescimento da primeira derivada é positiva, e que fará aumentar o valor da função objectivo e não diminui-lo. Conclui-se que os valores obtidos anteriormente são efectivamente os valores óptimos. De seguida serão apresentados os resultados para o uma divisão em 5 e em 10 elementos.

Para uma divisão em 5 elementos o problema de optimização seria:

$$\begin{cases} Min \quad f(x_1, x_2, x_3, x_4, x_5) = C_1(x_1 + x_2 + x_3 + x_4 + x_5) \\ \qquad s.a \begin{cases} \frac{1}{x_1^3} + \frac{7}{x_2^3} + \frac{19}{x_3^3} + \frac{37}{x_4^3} + \frac{61}{x_5^3} - C_2 \leq 0 \\ x_n > 0 \end{cases} \end{cases} \tag{A.27}$$

Com $C_1 = 4\rho L t$, $C_2 = \left(2tE \setminus FL^3\right) \delta_0$ e n o número de variáveis existentes (neste caso $n = 5$).

A função de Lagrange correspondente a este caso é apresentada em (A.28).

$$\mathcal{L} = C_1(x_1 + x_2 + x_3 + x_4 + x_5) + \lambda_1 \left(\frac{1}{x_1^3} + \frac{7}{x_2^3} + \frac{19}{x_3^3} + \frac{37}{x_4^3} + \frac{61}{x_5^3} - C_2\right) \tag{A.28}$$

Analogamente ao que foi feito para o caso da divisão da consola em 3 elementos conclui-se que os resultados serão:

$$\begin{cases} x_1^* = \left(\frac{1+7^{\frac{1}{4}}+19^{\frac{1}{4}}+37^{\frac{1}{4}}+61^{\frac{1}{4}}}{C_2}\right)^{\frac{1}{3}} \cong \frac{2.1527}{C_2^{\frac{1}{3}}} \\ x_2^* = 7^{\frac{1}{4}}\left(\frac{1+7^{\frac{1}{4}}+19^{\frac{1}{4}}+37^{\frac{1}{4}}+61^{\frac{1}{4}}}{C_2}\right)^{\frac{1}{3}} \cong \frac{3.5015}{C_2^{\frac{1}{3}}} \\ x_3^* = 19^{\frac{1}{4}}\left(\frac{1+7^{\frac{1}{4}}+19^{\frac{1}{4}}+37^{\frac{1}{4}}+61^{\frac{1}{4}}}{C_2}\right)^{\frac{1}{3}} \cong \frac{4.4943}{C_2^{\frac{1}{3}}} \\ x_4^* = 37^{\frac{1}{4}}\left(\frac{1+7^{\frac{1}{4}}+19^{\frac{1}{4}}+37^{\frac{1}{4}}+61^{\frac{1}{4}}}{C_2}\right)^{\frac{1}{3}} \cong \frac{5.3092}{C_2^{\frac{1}{3}}} \\ x_5^* = 61^{\frac{1}{4}}\left(\frac{1+7^{\frac{1}{4}}+19^{\frac{1}{4}}+37^{\frac{1}{4}}+61^{\frac{1}{4}}}{C_2}\right)^{\frac{1}{3}} \cong \frac{6.0160}{C_2^{\frac{1}{3}}} \end{cases} \quad \text{(A.29)}$$

Depois de efectuada a matriz Hessiana, que conclui que efectivamente o resultado obtido é o valor mínimo do problema, pode-se escrever a função objectivo que será:

$$Min \quad f(x_1, ..., x_n) = C_1 \sum_{i=1}^{n} x_i \cong \frac{21.4737 C_1}{C_2^{\frac{1}{3}}} \quad \text{(A.30)}$$

Com $C_1 = 4\rho L t$, $C_2 = \left(2tE \setminus FL^3\right)\delta_0$ e n o número de variáveis existentes (neste caso $n = 5$).
O mesmo processo foi feito dividindo a consola em 10 elementos. De seguida são apresentados os valores analíticos para as variáveis do problema.

$$x_1^* = \frac{3.0338}{C_1^{\frac{1}{3}}} \quad x_2^* = \frac{4.9347}{C_1^{\frac{1}{3}}} \quad x_3^* = \frac{6.3339}{C_1^{\frac{1}{3}}} \quad x_4^* = \frac{7.4823}{C_1^{\frac{1}{3}}} \quad x_5^* = \frac{8.4784}{C_1^{\frac{1}{3}}}$$

$$x_6^* = \frac{9.3701}{C_1^{\frac{1}{3}}} \quad x_7^* = \frac{10.1843}{C_1^{\frac{1}{3}}} \quad x_8^* = \frac{10.9384}{C_1^{\frac{1}{3}}} \quad x_9^* = \frac{11.6439}{C_1^{\frac{1}{3}}} \quad x_{10}^* = \frac{12.3090}{C_1^{\frac{1}{3}}} \quad \text{(A.31)}$$

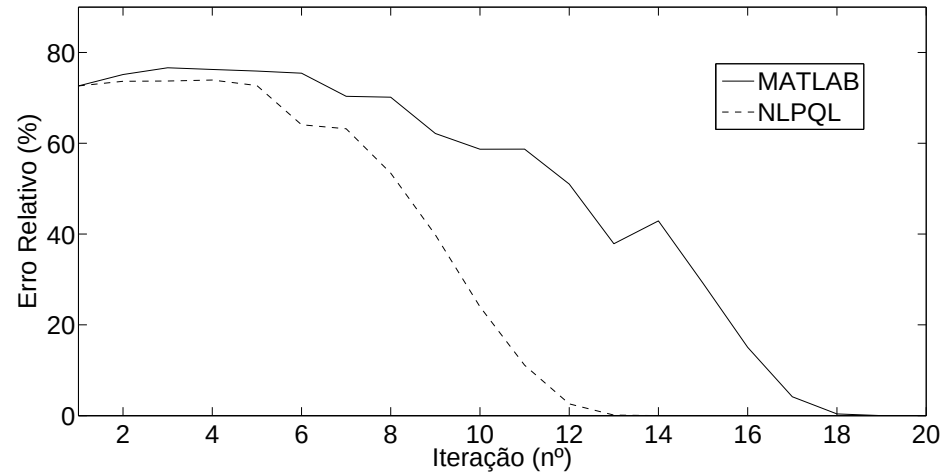
O valor obtido para a função objectivo foi:

$$Min \quad f(x_1, ..., x_n) = C_1 \sum_{i=1}^{n} x_i \cong \frac{84.7087 C_1}{C_2^{\frac{1}{3}}} \quad \text{(A.32)}$$

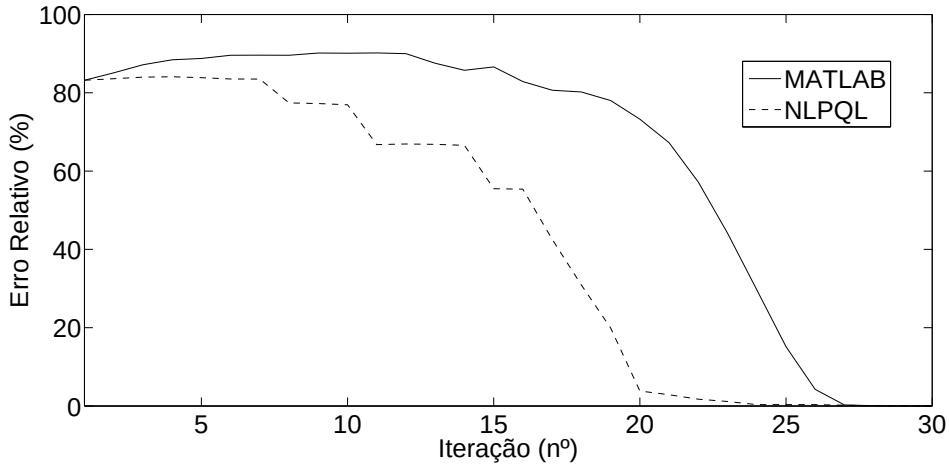
Os resultados obtidos através do algoritmo NLPQL e do software MATLAB são apresentados na Tabela A.1 e na Figura A.1

Nº Secções	Solução analítica (kg)	NLPQL		MATLAB	
		Nº iterações	Peso (kg)	Nº iterações	Peso (kg)
3	5163.1	18	5163.0	22	5163.1
5	14025.7	34	14025.4	39	14025.5
10	55327.1	64	55327.5	83	55327.4

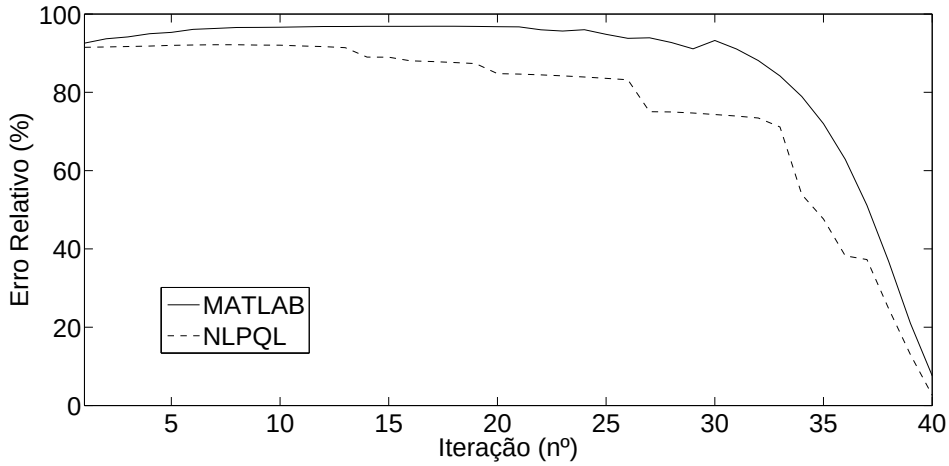
Tabela A.1: Optimização do peso da consola - Resultados obtidos para várias secções.



(a) Consola composta por 3 elementos.



(b) Consola composta por 5 elementos.



(c) Consola composta por 10 elementos.

Figura A.1: Evolução do erro durante o processo de optimização para vários elementos.

Apêndice B

Código utilizado nos problemas teste utilizando o programa NLPQL

A fim de provar que efectivamente o algoritmo NLPQL obtem um bom desempenho na resolução de problemas de optimização foi utilizado o seguinte código de programação para os problemas apresentados no capítulo 1.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccc MAIN PROGRAM ccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

PROGRAM MAIN

PARAMETER (NMAX = 100)
PARAMETER (MMAX = 1600)
PARAMETER (ME = 0)
PARAMETER (M = 1)
PARAMETER (LWA = NMAX*NMAX + MMAX*NMAX + 11*MMAX + 25*NMAX +
60)
PARAMETER (LKWA = 100)
INTEGER N,IPRINT,IFAIL,KWA(LKWA),MAXIT,MAXFUN,PROB
DOUBLE PRECISION X(NMAX),F,XL(NMAX),XU(NMAX),WA(LWA),EPS,SCBOU
COMMON /CMACHE/EPS

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccc IMPRIME PARA O ECRAN OS EXERCICIOS APRESENTADOS ccccccc
ccccccc NO CAPITULO 1 E O UTILIZADOR SELECIONA UM DELES ccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

PRINT *, '************************************************************'
PRINT *, '*************CHOOSE THE BENCHMARK PROBLEM:********************'
```

```
PRINT *, '************************************************************'
PRINT *, 'MINIMIZATION OF THE CANTILEVER WITH 2 ELEMENTS—> write: 1'
PRINT *, 'MINIMIZATION OF THE CANTILEVER WITH 3 ELEMENTS—> write: 2'
PRINT *, 'MINIMIZATION OF THE CANTILEVER WITH 5 ELEMENTS—> write: 3'
PRINT *, 'MINIMIZATION OF THE CANTILEVER WITH 10 ELEMENTS–> write:
4'
PRINT *, 'MAXIMIZATION OF THE STRUCTURE STIFFNESS—> write: 5'
PRINT *, '************************************************************'
READ (*,*)PROB

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccc CONFORME A ESCOLHA DO UTILIZADOR ATRIBUI-SE ccccccccc
ccccccccccccccc O NÚMERO DE VARIÁVEIS DO PROBLEMA cccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

IF (PROB.EQ.1) N=2
IF (PROB.EQ.2) N=3
IF (PROB.EQ.3) N=5
IF (PROB.EQ.4) N=10
IF (PROB.EQ.5) N=3

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccc SÃO DEFINIDOS ALGUNS PARÂMETROS NECESSÁRIOS ccccccc
ccccccccccccccccc PARA A EXECUÇÃO DA OPTIMIZAÇÃO cccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

EPS=1.D-3
SCBOU=1.D+3
MAXFUN=100
MAXIT=3000
IFAIL = 0
IPRINT=2
IOUT=21

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccc GUARDA-SE NUM FICHEIRO O NÚMERO cccccccccccccccc
ccccccccccc DO PROBLEMA ESCOLHIDO PARA SER ABERTO ccccccccccccc
ccccccccccccccccc NA SUBROTINA NLFUNC E NLGRAD ccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

OPEN(10,file='DATA.dat',status='unknown')
WRITE(10,*) PROB
CLOSE(10)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccc ATRIBUI-SE OS LIMITES DAS VARIÁVEIS cccccccccccccccc
ccccccccccccccccccccccc E OS VALORES INCIAIS ccccccccccccccccccccccc
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

DO I=1,N
X(I)=0.15D0
XL(I)=1D-4
XU(I)=100D0
ENDDO

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccc CHAMA-SE A SUBROTINA PARA COMEÇAR A OPTIMIZAÇÃO ccccccc
ccccccccc E NO FIM IMPRIME-SE PARA O ECRAN OS RESULTADOS ccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

OPEN(IOUT,file='OUT.dat',status='unknown')

CALL      NLPQL0(M,ME,N,X,F,XL,XU,IPRINT,IFAIL,WA,LWA,KWA,LKWA,MAXIT,
MAXFUN,SCBOU)
WRITE(*,*)
WRITE(*,*)'NPQL FINAL RESULTS'
WRITE(*,*)'******************'
WRITE(*,*)'IFAIL', IFAIL
WRITE(*,*)'Design variables:'
DO i = 1, N
WRITE(*,"(' ','X(',i2,') = ',E15.9)") I,X(I)
ENDDO
WRITE(*,*)'Objective function:'
WRITE(*,"(' ','F = ',E15.9)") F
WRITE(*,*)
IF (IFAIL .eq. 0) then
WRITE(*,*)'— NORMAL TERMINATION OF THE NLPQL OPTIMIZER —'
ELSE
WRITE(*,*)'— ANORMAL TERMINATION OF THE NLPQL OPTIMIZER —'
ENDIF
WRITE(*,*)'************************************************'
END



ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccc NLFUNC ccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

SUBROUTINE NLFUNC(M,ME,MMAX,N,F,G,X,ACTIVE)

INTEGER PROB,N
LOGICAL ACTIVE(2*MMAX+15)
DOUBLE PRECISION X(N),F,G(M)
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccc ABRE O FICHEIRO E LÊ O PROBLEMA ccccccccccccccc
ccccccccccccccccc ESCOLHIDO NO MAIN PROGRAM ccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

OPEN(10,file='DATA.dat',status='unknown')
READ(10,*) PROB
CLOSE(10)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccc DEFINIÇÃO DAS FUNÇÕES DO PROBLEMA ccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

IF (PROB.NE.5) THEN
ccccccccccccc EXERCICIO PESO CONSOLA (n=2) ccccccccccccccccc
IF (N.EQ.2) THEN
F=3140D0*(X(1)+X(2))
G(1)=111.111D0-(1/X(1)**3)-(7/X(2)**3)
end if
ccccccccccccc EXERCICIO PESO CONSOLA (n=3) ccccccccccccccccc
IF (N.EQ.3) THEN
F=3140*(X(1)+X(2)+X(3))
G(1)=111.111D0-(1/X(1)**3)-(7/X(2)**3)-(19/X(3)**3)
end if
ccccccccccccc EXERCICIO PESO CONSOLA (n=5) ccccccccccccccccc
IF (N.EQ.5) THEN
F=3140*(X(1)+X(2)+X(3)+X(4)+X(5))
G(1)=111.111D0-(1/X(1)**3)-(7/X(2)**3)-(19/X(3)**3)-(37/X(4)**3)-(61/X(5)**3)
end if
ccccccccccccc EXERCICIO PESO CONSOLA (n=10) ccccccccccccccccc
IF (N.EQ.10)THEN
F=3140*(X(1)+X(2)+X(3)+X(4)+X(5)+X(6)+X(7)+X(8)+X(9)+X(10))
G(1)=111.111D0-(1/X(1)**3)-(7/X(2)**3)
/ -(19/X(3)**3)-(37/X(4)**3)-(61/X(5)**3)-(91/X(6)**3)
/ -(127/X(7)**3)-(169/X(8)**3)-(217/X(9)**3)-(271/X(10)**3)
end if

ENDIF
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccc EXERCICIO MAXIMO RIGIDEZ ccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 F= 0.0083*(1/X(1)+1/X(2)+(2*(2**0.5))/X(3))
G(1)=1.0-X(1)-X(2)-(2**0.5)*X(3)
```

```
END


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccc NLGRAD ccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

SUBROUTINE NLGRAD(M,ME,MMAX,N,F,G,DF,DG,X,ACTIVE,WA)

PARAMETER(MMAX1=1600)
INTEGER N,M,PROB
DOUBLE PRECISION X(N),F,WA(MMAX),DG(MMAX,N),DF(N),EPS
DOUBLE PRECISION G(MMAX)
LOGICAL ACTIVE(2*MMAX+15)
COMMON /CMACHE/EPS
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccc ABRE O FICHEIRO E LÊ O PROBLEMA ccccccccccccccc
ccccccccccccccccccc ESCOLHIDO NO MAIN PROGRAM ccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

OPEN(10,file='DATA.dat',status='unknown')
READ(10,*) PROB
CLOSE(10)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccc DEFINIÇÃO DAS DERIVADAS PARCIAS ccccccccccccccc
ccccccccccccccccccccccc DAS FUNÇÕES DO PROBLEMA ccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

cccccccc PARA O EXERCICIO DA IF (PROB.NE.5) THEN
DO J=1,N
DF(J)=3140
ENDDO
DG(1,1)=3/X(1)**4
DG(1,2)=21/X(2)**4
IF (N.EQ.2) GOTO 1
ccccccccccccccc PÁRA AQUI PARA O CASO DA CONSOLA (N=2) ccccccc
DG(1,3)=57/X(3)**4
IF (N.EQ.3) GOTO 1
ccccccccccccccc PÁRA AQUI PARA O CASO DA CONSOLA (N=3) ccccccc
DG(1,4)=111/X(4)**4
DG(1,5)=183/X(5)**4
IF (N.EQ.5) GOTO 1
ccccccccccccccc PÁRA AQUI PARA O CASO DA CONSOLA (N=5) ccccccc
DG(1,6)=273/X(6)**4
DG(1,7)=381/X(7)**4
DG(1,8)=507/X(8)**4
```

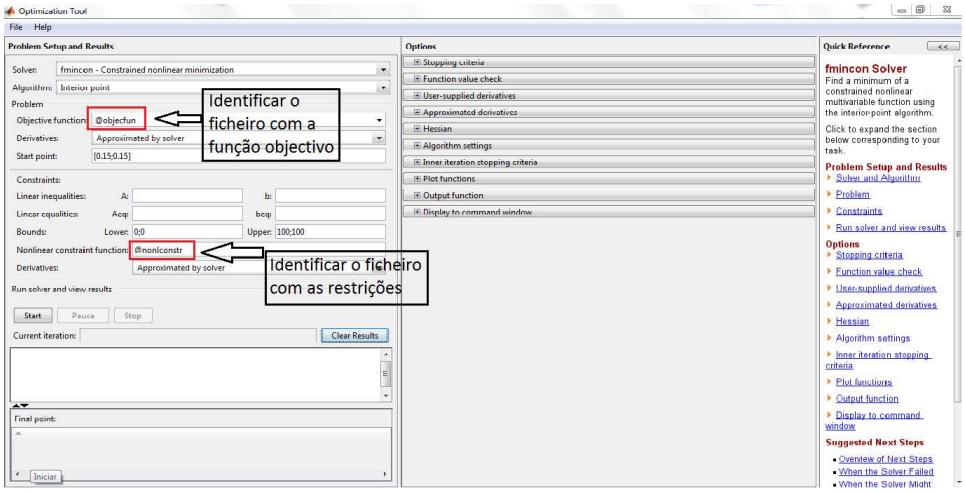
```
DG(1,9)=651/X(9)**4
DG(1,10)=813/X(10)**4

ENDIF
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccccccccccccccc EXERCICIO MAXIMO RIGIDEZ ccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

DF(1)=-0.0083*(1/X(1)**2)
DF(2)=-0.0083*(1/X(2)**2)
DF(3)=-0.0083*((2*(2**0.5))/X(3)**2)

DG(1,1)=-1
DG(1,2)=-1
DG(1,3)=-2**0.5
1 CONTINUE
END
```

Apêndice C

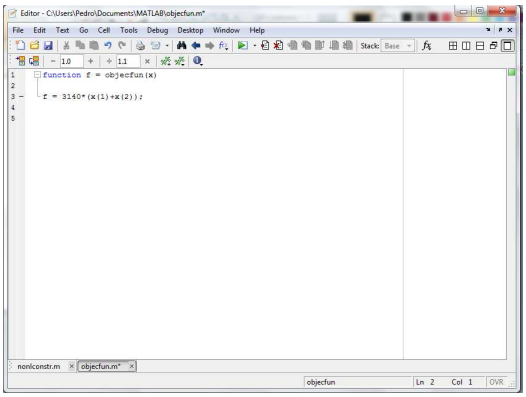
Optimização de problemas recorrendo ao software MATLAB

Para a resolução dos problemas teste apresentados no Capítulo 2 foi utilizado o software Matlab. Este programa de cálculo possibilita a resolução de problemas de optimização recorrendo a diversos algoritmos, dos quais foi selecionado o *Interior-Point Algorithm*.

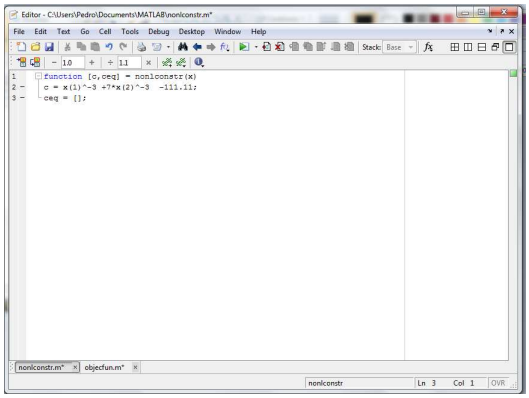
Para se proceder à execução da optimização, o utilizador deverá na linha de comando escrever **optimtool**. Ao accionar esta opção aparece no ecran um quadro semelhante ao representado na Figura C.1. É neste quadro que o utilizador define todos os parâmetros referentes ao problema de optimização. Pode-se verificar que nas primeiras opções o utilizador poderá escolher qual o algoritmo que deseja utilizar ou quais os intervalos a que estão restringidas as variáveis. As funções integrantes do problema são colocadas em dois ficheiros desenvolvidos à parte. Num dos ficheiros deverá colocar a função objectivo e no outro as restrições. O utilizador deverá indicar, conforme apresentado na Figura C.1, qual o ficheiros que contêm as funções [21].



(a) Quadro Optimtool



(b) Código com a função objectivo.



(c) Código com as restições do problema.

Figura C.1: Ferramenta Optimtool, disponível no software MATlab.

Apêndice D

Ficheiro tipo a utilizar no algoritmo HTD

Para a execução do algortimo HTD é necessário o utilizador proceder a execução de um ficheiro com a informação sobre a geometria da estrutura, carregamento e parâmetros necessários para a sua execução. De seguida é apresentado o ficheiro utilizado no primeiro exercício do capítulo 3. Este ficheiro pode variar consoante o tipo de estrutura e o tipo de carregamento, mas as suas variações não são muito significativas.

```
TITLE OF THE PROBLEM: CANTILEVER 1E
*NUMBER OF NODES
4
*COORDINATES OF THE POINTS
1 0.0 0.0
2 1.0 0.0
3 1.0 1.0
4 0.0 1.0
*NUMBER OF ELEMENTS
1
*ELEMENT1, NUMBER OF SIDES
4 0 0 0 0
*SIDE , GENERATING POINTS (4) AND TYPE
1 1 2 0 0 0
2 2 3 0 0 0
3 3 4 0 0 0
4 4 1 0 0 1
*NUMBER OF LOADED SIDES
1
*ELEM,SIDE,GEN.POINTS FOR TN, GEN.POINTS FOR TT
1 2 1 0
*Tn
1.0
```

```
*NUMBER OF SIDES WITH IMPOSED DISPLACEMENTS
1
* ELEM,SIDE,FLAG...
1 4 0
*GEN.POINTS FOR UN (NR., POINTS)
1 0
*GEN.POINTS FOR UT (NR., POINTS)
1 0
*PARAMETERS: E,MIU,OMEGA,A,AT
1 0.3 6.28318530717958 18 7
*NUMBER OF SINGULARITY POINTS
0
*NR.OF IMPOSED NODAL DISPLACEMENTS
0
*NR.OF DIVISIONS
50
*OUTPUT KODE (0=NO OUTPUT, 1=SHORT OUTPUT, 2=LONG OUTPUT)
0
*PIVOTING KODE (0="BEST"PIVOTING, 1=IMPOSE DIAGONAL PIVOTING)
0
* Cod concentrated load code ("0"no concentrated load)
0
* Solver (H:Harwell, S:Sparse, G:Gauss, I:Iterative CGM)
S
* Scaling code (1:Harwell, 2:our scaling)
1
* Tractions approximation basis (N:normal polynomials, C:Chebyshev)
C
```

Os resultados são apresentados num ficheiro chamado $STRESS\ TABLE$ que apresenta a seguinte configuração:

```
ELEM.   X    Y      SIGMA XX    SIGMA YY    SIGMA XY

1       0.250   0.250   0.1013E+02   0.6606E+00   0.3228E+00
```

Apêndice E

Código utilizado no algoritmo de optimização

Para a realização deste trabalho procedeu-se a execução de um programa que permitisse a optimização das estruturas recorrendo a elementos finitos híbridos de Trefftz. Neste apêndice é apresentado o código desenvolvido para resolver o processo apresentado aquando da explicação do funcionamento do programa. O código aqui transcrito encontra-se pronto para executar a optimização da tensão normal ao eixo x efectuado na consola que foi alvo de estudo neste trabalho. Relembra-se que este programa foi desenvolvido na linguagem de programação Fortran.
Partindo do que foi explicado no subcapitulo 4 passa-se a apresentar os seguintes códigos:

- Dados.out
- Programa HTD
- Pós processamento
- Get tensoes
- Get area
- Get desl
- MAIN PROGRAM
- NLFUNC
- NLGRAD

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 ccccccccccccccccccccccccc DADOS.OUT ccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


program dados
DOUBLE PRECISION X(3)

ccccc Começa por abrir o ficheiro Xi onde se encontram guardados cccc
ccccc os valores das variáveis e guarda-os de forma a que se possa depois cccc
ccccc escrever o ficheiro de entrada do programa HTD cccc

OPEN(40,file='Xi')

DO i=1,3
READ(40,*)X(i)
END DO
CLOSE(40)

ccccc Começa a escrever o ficheiro de texto e apenas altera-o cccc
ccccc nos locais onde se encontram as variáveis do problema cccc

open(1,file='dadostexto')

100 format(A)
60 format(I2,x,F8.5,x,F8.5)
70 format(I2,x,I2,x,I2,x,F8.5,x,F8.5,x,I2)

write(1,100)'TITLE OF THE PROBLEM:DADOS'
write(1,100)'*NUMBER OF NODES'
write(1,100)'4'
write(1,100)'*COORDINATES OF THE POINTS'
write(1,100)'1 0.0 0.0'
write(1,60)2,4.0,X(1)    cccc escreve aqui X(1)
write(1,100)'3 4.0 2.0'
write(1,100)'4 0.0 2.0'
write(1,100)'*NUMBER OF ELEMENTS'
write(1,100)'1'
write(1,100)'*ELEMENT1, NUMBER OF SIDES'
write(1,100)'4 6 0 0 0'
write(1,100)'*SIDE , GENERATING POINTS (4) AND TYPE'
write(1,70)1,1,2,X(2),X(3),0    cccc escreve aqui X(2) e X(3)
write(1,100)'2 2 3 0 0 0'
write(1,100)'3 3 4 0 0 0'
write(1,100)'4 4 1 0 0 1'
write(1,100)'*NUMBER OF LOADED SIDES'
```

```
write(1,100)’1’
write(1,100)’* ELEM,SIDE,GEN.POINTS FOR TN , GEN.POINTS FOR TT’
write(1,100)’1 3 1 0’
write(1,100)’*Tn’
write(1,100)’-0.1’
write(1,100)’* NUMBER OF SIDES WITH IMPOSED DISPLACEMENTS’
write(1,100)’1’
write(1,100)’* ELEM,SIDE,FLAG...’
write(1,100)’1 4 0’
write(1,100)’*GEN.POINTS FOR UN (NR., POINTS)’
write(1,100)’1 0’
write(1,100)’*GEN.POINTS FOR UT (NR., POINTS)’
write(1,100)’1 0’
write(1,100)’*PARAMETERS: E,MIU,OMEGA,A,AT’
write(1,100)’1 0.3 6.28318530717958 7 1’
write(1,100)’*NUMBER OF SINGULARITY POINTS’
write(1,100)’0’
write(1,100)’*NR.OF IMPOSED NODAL DISPLACEMENTS’
write(1,100)’0’
write(1,100)’*NR.OF DIVISIONS’
write(1,100)’50’
write(1,100)’*OUTPUT KODE (0=NONE, 1=SHORT, 2=LONG)’
write(1,100)’2’
write(1,100)’*PIVOTING KODE(0="BEST",1=IMPOSE DIAGONAL PIVOTING)’
write(1,100)’0’
write(1,100)’*Cod concentrated load code("0"no concentratedload)’
write(1,100)’0’
write(1,100)’*Solver(H:Harwell,S:Sparse,G:Gauss,I:Iterative CGM)’
write(1,100)’S’
write(1,100)’*Scaling code (1:Harwell, 2:our scaling)’
write(1,100)’1’
write(1,100)’*Tractions approximation(N:normal,C:Chebyshev)’
write(1,100)’C’

close(1)
stop
end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 cccccccccccccccccccccc Programa HTD ccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

dados.out
./prog «__
dadostexto
OUTPUT
Y
__


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 cccccccccccccccccccc Pós processamento ccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

../program/g « __
N
Y
1
1,1,7
N
N
N
__


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 cccccccccccccccccccccccccc Get Tensoes ccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

program main

Integer s, div(100), points,gg1,gg2,gg3
Character Q ,Q2

points=0;
s=0;

cccc Para saber o número de pontos definidos pelo utilizador cccc
cccc abre o ficheiro Run G e processa a sua leitura cccc

open(30,file='run_g')
read(30,*)
read(30,*)
read(30,*)Q
```

```
If (Q.eq.'Y') then
read(30,*)s
do k=1,s
read(30,*)el,side,div(k)
enddo
end if
read(30,*)Q
If (Q.eq.'Y') then
read(30,*)points
end if
close(30)

cccc Sabendo o número de lados e divisões pedidas pelo utilizador cccc
cccc sabe-se o número de pontos e assim sendo recorre-se a um ciclo cccc
cccc para guardar os valores das tensões no ficheiro 'tensor' cccc

open(10,file='STRESS_TABLE')
open(20,file='tensor')

if (s.NE.0) then
read(10,*)
read(10,*)
do gg2=1,s
do gg1=1,div(gg2)*3+3 ccc Para fronteiras lineares do gg1=1,div(gg2)+1
read(10,*)el,side,x,y,sigxx,sigyy,sigxy
write(20,*)sigxx
write(20,*)sigyy
write(20,*)sigxy
enddo
enddo
endif

if(points.NE.0) then
read(10,*)
read(10,*)
read(10,*)
do gg3=1,points
read(10,*)el,x,y,sigxx,sigyy,sigxy
write(20,*)sigxx
write(20,*)sigyy
write(20,*)sigxy
enddo
endif
close(10)
close(20)
stop
end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 ccccccccccccccccccccccccccc Get Area ccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

program main

Integer elem,nodes
Character A
double precision ar1

cccc Procura nos ficheiros criados os valores das areas ccccc

call system('grep SURFACE el* » exp_grep')

cccc Vou procurar no ficheiro que dou de entrada qual cccc
cccc é o número de elementos usados cccc

area=0;
open(3,file='dadostexto')
read(3,*)
read(3,*)
read(3,*)nodes
read(3,*)
do i=1,nodes
read(3,*)
end do
read(3,*)
read(3,*)elem
close(3)

ccccccc Fez-se um ciclo que me guarda o valor de cada linha (cada elemento) ccc
cccccccc e vou somando, no fim obtenho a area total da estrutura ccccccccccc

100 format(A,A,A,A,A,A,A,A,F8.5)
open(10,file='exp_grep')
open(20,file='area_dos_elementos')
do i=1,elem
read(10,100)B,B,B,B,B,B,B,B,ar1
write(20,*)ar1
end do
close(10)
close(20)

cccc Com a informação obtida no ficheiro area_dos_elementos obtém-se a ccc
cccc àrea em cada elemento (caso se pretenda outro tipo de optimização) ccc
cccc que depois somando obtem-se a àrea total da estrutura ccc
```

```
open(40,file=’area_dos_elementos’)
do i=1,elem
read(40,*)x
area=x+area;
end do
close(40)

ccccc Escreve-se o valor da àrea num ficheiro a ser cccc
ccccc utilizado na definição das funções do problema (subrotina NLFUNC ) cccc

open (5,file=’areatot’)
write(5,*)area
close(5)

cccccc Apaga-se os ficheiros que não fazem falta ccccccccccccccccccccccccc

call system(’rm exp_grep’)
call system("rm area_dos_elementos")

end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 ccccccccccccccccccccccccccccc Get desl ccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

program main

Integer div(100),s,points,deslocamentos,gg1,gg2
Character Q

deslocamentos=0;
points=0;
s=0;
cccc Para saber o número de pontos definidos pelo utilizador cccc
cccc abre o ficheiro Run G e processa a sua leitura cccc

open(30,file=’run_g’)
read(30,*)
read(30,*)
read(30,*)Q
If (Q.eq.’Y’) then
read(30,*)s
do k=1,s
read(30,*)
enddo
end if
```

```
read(30,*)Q
If (Q.eq.'Y') then
read(30,*)points
do k=1,points
read(30,*)
enddo
end if
read(30,*)Q
If (Q.eq.'Y') then
read(30,*)deslocamentos
do k=1,deslocamentos
read(30,*)el,side,div(k)
enddo
end if
close(30)

cccc Sabendo o número de lados e divisões pedidas pelo utilizador cccc
cccc sabe-se o número de pontos e assim sendo recorre-se a um ciclo cccc
cccc para guardar os valores do deslocamento no ficheiro 'DESL' cccc


open(10,file='DISPLACEMENTS_TABLE')
open(20,file='DESL')
if (deslocamentos.NE.0) then
read(10,*)
read(10,*)
do gg2=1,deslocamentos
do gg1=1,div(gg2)+1
read(10,*)el,side,x,y,dx,dy
write(20,*)dx
write(20,*)dy
enddo
enddo
endif

close(10)
close(20)
stop
end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 ccccccccccccccccccccccccc MAIN PROGRAM ccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


PROGRAM main

PARAMETER (NMAX = 100)
PARAMETER (MMAX = 1600)
PARAMETER (ME = 0)
PARAMETER (M = 23)
PARAMETER (N = 3)
PARAMETER (LWA = NMAX*NMAX + MMAX*NMAX + 11*MMAX + 25*NMAX +
60)
PARAMETER (LKWA = 100)
INTEGER IPRINT,IFAIL,KWA(LKWA),MAXIT,MAXFUN
DOUBLE PRECISION X(NMAX),F,XL(NMAX),XU(NMAX),WA(LWA),EPS,SCBOU
COMMON /CMACHE/EPS

ccccccc Parâmetros ccccccccccccccccccccccccccccccccc

EPS=1.D-3
SCBOU=1.D+3
MAXFUN=100
MAXIT=3000
IFAIL = 0
IPRINT=3
IOUT=21

ccccccc Definição dos limites e valores iniciais ccccccccccccc

X(1)=0D0
XL(1)=-1.0
XU(1)=1.8

X(2)=0D0
XL(2)=-6.28
XU(2)=6.28

X(3)=0D0
XL(3)=-6.28
XU(3)=6.28

ccccccc Criação do ficheiro de saída e da informação cccccccccccccc
ccccccc a ser disponibilizada no ecrãn cccccc
```

```
OPEN(IOUT,file='OUT')

CALL NLPQL0(M,ME,N,X,F,XL,XU,IPRINT,IFAIL,WA,LWA,KWA,LKWA,MAXIT,
/MAXFUN,SCBOU)
WRITE(*,*)
WRITE(*,*)'NPQL FINAL RESULTS'
WRITE(*,*)'******************'
WRITE(*,*)'IFAIL', IFAIL
WRITE(*,*)'Design variables:'
DO i = 1, N
WRITE(*,"(' ','X(',i2,') = ',E15.9)") I,X(I)
ENDDO
WRITE(*,*)'Objective function:'
WRITE(*,"(' ','F = ',E15.9)") F
WRITE(*,*)
IF (IFAIL .eq. 0) then
WRITE(*,*)'— NORMAL TERMINATION OF THE NLPQL OPTIMIZER —'
ELSE
WRITE(*,*)'— ANORMAL TERMINATION OF THE NLPQL OPTIMIZER —'
ENDIF
WRITE(*,*)'************************************************'
END

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 ccccccccccccccccccccccccccc NLFUNC ccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


SUBROUTINE NLFUNC(M,ME,MMAX,N,F,G,X,ACTIVE)

INTEGER N
LOGICAL ACTIVE(2*MMAX+15)
DOUBLE PRECISION X(N),F,G(M),A0,T0x,T0y

cccc Cria-se o ficheiro Xi com a informação do valor das variáveis cccc

OPEN(30,file='Xi')
DO I=1,N
WRITE(30,*)X(I)
END DO
CLOSE(30)

cccc Executa o programa HTD e recolhe a informação das tensões cccc
cccc nos pontos, dos deslocamentos e da área cccc

call system ('rm STRESS_TABLE')
call system ('run_prog > lixo')
```

```
call system (’run_g > lixo’)
call system (’get_area’)
call system (’get_tensoes’)
call system (’get_desl’)


OPEN(80,file=’areatot’)
READ(80,*)A0
CLOSE(80)

F=A0;

cccc E por outro lado cria o ficheiro ’A0T0’ onde é guardada cccc
cccc a informação a ser usada na subrotina NLGRAD cccc

OPEN(70,file=’tensor’)
open(120,file=’A0T0’)
write(120,*)A0

cccc Antes da definição das restrições procede-se a leitura cccc
cccc do ponto singular sem que seja guardado qualquer valor cccc

READ(70,*)T0x
READ(70,*)T0Y
READ(70,*)T0xy


DO ii=1,M

READ(70,*)T0x
READ(70,*)T0Y
READ(70,*)T0xy

WRITE(120,*)T0x

if (ACTIVE(ii)) then
G(ii)=-ABS(T0x)+2
end if

ENDDO
RETURN
END
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
 ccccccccccccccccccccccccccc NLGRAD ccccccccccccccccccccccccccccccccccc
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


SUBROUTINE NLGRAD(M,ME,MMAX,N,F,G,DF,DG,X,ACTIVE,WA)

PARAMETER(MMAX1=1600)
INTEGER N,M
DOUBLE PRECISION X(N),F,WA(MMAX),DG(MMAX,N),DF(N),EPS
DOUBLE PRECISION G(MMAX),DELTA,A0,T0x,Tx,A,Ty,T0y
LOGICAL ACTIVE(2*MMAX+15)
COMMON /CMACHE/EPS

DELTA=0.01;

do jj=1,N

cccc Cria-se o ficheiro Xi com a informação do valor das variáveis cccc
cccc já com o valor delta acrescentado á variável em causa cccc

OPEN(30,file='Xi')
DO I=1,N
if (i.eq.jj) then
WRITE(30,*)X(I)+delta
else
WRITE(30,*)X(I)
endif
END DO
CLOSE(30)

cccc Executa o programa HTD e recolhe a informação das tensões cccc
cccc nos pontos, dos deslocamentos e da área cccc


call system ('rm STRESS_TABLE')
call system ('run_prog > lixo')
call system ('run_g > lixo')
call system ('get_area')
call system ('get_tensoes')
call system ('get_desl')


OPEN(60,file='areatot')
READ(60,*)A
CLOSE(60)
```

```
cccc Abre o ficheiro A0T0 com a informação obtida na subrotina cccc
cccc NLFUNC e define a derivadas parciais para a variável em causa cccc

OPEN(120,file='A0T0')
READ(120,*)A0
DF(jj)=(A-A0)/DELTA;

OPEN(70,file='tensor')

cccc Antes da definição das restrições procede-se a leitura cccc
cccc do ponto singular sem que seja guardado qualquer valor cccc

READ(70,*)Tx
READ(70,*)Ty
READ(70,*)Txy

DO ii=1,M

READ(70,*)Tx
READ(70,*)Ty
READ(70,*)Txy

READ(120,*)T0x

if(ACTIVE(ii))then
DG(ii,jj)=(Tx-T0x)/DELTA;
endif
END DO

close(70)
close(120)

ENDDO
RETURN
END
```

Apêndice F

Programa de optimização - Instruções de uso

O utilizador para que possa usar o programa de optimização desenvolvido no presente trabalho necessita de seguir os passos apresentados de seguida. Relembra-se que o programa foi desenvolvido em linguagem de programação Fortran. Os códigos apresentados no **apêndice E** encontram-se programados para executar a optimização da consola limitando a fronteira inferior a um valor de $\sigma_{xx} = -2.0kPa$ (subcapítulo 4.8.1).

- Primeiro o utilizador tem de compilar algumas funções já apresentadas neste trabalho. São elas a $get\_tensoes$, a get_desl e a get_area. O código associado a estas funções podem ser consultados nas páginas 92, 94 e 95;
- De seguida o utilizador terá de desenvolver a função que actualize o ficheiro de entrada. Um exemplo deste ficheiro é apresentado na página 90. O utilizador deverá identificar quais as variáveis conforme é apresentado no exemplo. No final deverá compilar a função com o nome $dados.out$;
- O utilizador deverá copiar para um ficheiro a função Programa HTD (página 92). O nome deste ficheiro deverá ser run_prog;
- Posteriormente, o utilizador terá de desenvolver a função pós processamento. É neste ficheiro onde se especifica onde se pretende determinar as tensões ou os deslocamentos. Um exemplo deste ficheiro é apresentado no na página 92. É aconselhável que o utilizador recorra à Figura 3.3 na elaboração este ficheiro. No final o nome do ficheiro deverá ser run_g;
- A subrotina NLGRAD apresentada no **apêndice E** não deverá sofrer nenhuma alteração. O utilizador deverá somente copiar o código fornecido. Em relação à subrotina NLFUNC, o utilizador necessita apenas de indicar quais os valores máximos que as restrições poderão tomar. Para tal apenas terá de introduzir o respectivo valor na linha de código ($G(ii) = -ABS(T0x) + 2$). Neste exemplo o valor limite da restrição é de 2. O código relativo a estas subrotinas encontram-se nas páginas 98 e 100;

- Por fim, é necessário que o utilizador indique o número de restrições, o número de variáveis e os seus limites. No exemplo apresentado na página 97 verifica-se que foram consideradas três variáveis ($PARAMETER\,(N\,=\,3)$) e vinte e três restrições ($PARAMETER\,(M\,=\,23)$). Igualmente se poderá verificar que a primeira variável está restringida ao intervalo $[-1.0; 1.8]$ e as restantes a $[-2\pi; 2\pi]$.

Para executar o programa de optimização o utilizador deverá compilar todas as subrotinas pertencentes ao algoritmo NLPQL. São elas: a $NLFUNC$, $NLGRAD$, QL, $MERIT$, $LINSEA$ e o $Main\ Program$. No final do processo de optimização os resultados ficarão guardados no ficheiro OUT.